```
        TITLE  "TURTLE/WSFN"

;
; TURTLE GRAPHICS PROCESSOR FOR ATARI CANDY/COLLEEN
;
; EDIT #28 -- MARCH 18, 1981
;
; BASED IN PART UPON 'WSFN' BY LICHEN WANG AS DESCRIBED IN
; DDJ NUMBER 18. DEVELOPED BY HARRY B. STEWART 1978,1979.
;
```

Cartridge
Cassette
Disk

```
        TITLE  "TURTLE/WSFN"

; TURTLE GRAPHICS PROCESSOR FOR ATARI CANDY/COLLEEN

; EDIT #28 -- MARCH 18, 1981

; BASED IN PART UPON 'WSFN' BY LICHEN WANG AS DESCRIBED IN
; DDJ NUMBER 18  DEVELOPED BY HARRY B  STEWART 1978,1979.
```

Cartridge version = CTRTLY.OBJ
Cassette version = BTRTLY.OBJ 0700-1B9A
Disk load version = LTRTLY.OBJ 3000-468C

```
; TURTLE GRAPHICS PROGRAM EQUATES
;
;  EQUATE FOR BOOT VERSION OR CARTRIDGE VERSION
;
; BOOT = 0 & BLOAD = 0 PRODUCES THE CARTRIDGE VERSION
; BOOT = 1 & BLOAD = 0 PRODUCES THE CASSETTE/DISK BOOTABLE VERSION
; BOOT = 1 & BLOAD = 1 PRODUCES THE DISK LOADABLE VERSION
; BOOT = 0 & BLOAD = 1 IS NOT A VALID COMBINATION!
;
BLOAD   = 0             ;DISK LOAD = 1, BOOT = 0
BOOT    = 1             ;BOOTABLE = 1, CARTRIDGE = 0
;
;    COLLEEN I/O
;
CIO     = $E456
;
IOVBAS  = $E400         COLLEEN VECTOR BASE ADDRESS
EPUTC   = $E406         "E " PUT CHARACTER
SGETC   = $E414         "S " GET CHARACTER
SPUTC   = $E416         "S " PUT CHARACTER
;
IOCBSZ  = 16            16 BYTES PER IOCB
IOCB0   = $00           TEXT OUTPUT
IOCB1   = IOCB0+IOCBSZ  TEXT INPUT
IOCB2   = IOCB1+IOCBSZ  GRAPHICS INPUT & OUTPUT
IOCB3   = IOCB2+IOCBSZ  GET/PUT USER DEFINITIONS
IOCB4   = IOCB3+IOCBSZ  (UNUSED)
;
ICHID   = $0340         IOCB HANDLER I.D.
ICDNO   = ICHID+1       DEVICE #
ICCOM   = ICDNO+1       COMMAND BYTE
ICSTA   = ICCOM+1       STATUS BYTE
ICBAL   = ICSTA+1       BUFFER ADDRESS (LO)
ICBAH   = ICBAL+1       BUFFER ADDRESS (HI)
ICRLL   = ICBAH+1       RECORD LENGTH (LO)
ICRLH   = ICRLL+1       RECORD LENGTH (HI)
ICBLL   = ICRLH+1       BUFFER LENGTH (LO)
ICBLH   = ICBLL+1       BUFFER LENGTH (HI)
ICAUX1  = ICBLH+1       AUX1
ICAUX2  = ICAUX1+1      AUX2
;
OPEN    = $03           OPEN COMMAND
CLOSE   = $0C           CLOSE COMMAND
GETC    = $07           GET CHARACTER COMMAND
PUTC    = $0B           PUT CHARACTER COMMAND
;
OREAD   = $04           OPEN DIRECTION
OWRIT   = $08           OPEN DIRECTION
SPLIT   = $10           SPLIT SCREEN OPTION
NOCLR   = $20           INHIBIT SCREEN CLEAR OPTION
;
DEL     = $7E           USER COMMAND DELETE CHARACTER (INTERNAL)
EOF     = $1A           INTERNAL END-OF-FILE CHARACTER (CTRL-Z)
EOL     = $9B           ATASCII CARRIAGE RETURN
CLEAR   = $7D           MONITOR CLEAR SCREEN
BELL    = $FD           BELL CODE
DELCH   = $FE           DELETE CHARACTER CODE
;
;    SIZING PARAMETERS
;
LINSIZ  = 80            # OF CHARACTERS PER SCREEN LINE (TEXT)
ULINES  = 12            # OF LINES OF USER DEFINED FUNCTIONS
VLINES  = 1             # OF LINES OF USER DEFINED VARIABLES
USIZE   = ULINES*LINSIZ
VSIZE   = VLINES*LINSIZ
VLNGTH  = 8             # OF BYTES PER VARIABLE DEFINITION (MUST BE >= NL + 2)
```

```
NUMFNS  = 12            # OF LINES OF USER DEFINED FUNCTIONS
NUMVRS  = 1             # OF LINES OF USER DEFINED VARIABLES
VBUFF   = NUMFNS+NUMVRS
VBUFE   = VBUFF+NUMFNS+1
VLNGTH  = 8             # OF BYTES PER VARIABLE DEFINITION (MUST BE >= ML + 2)

DNSIZE  = 15            DEVICE/FILENAME MAXIMUM SIZE
ML      = 4             # OF DIGITS IN NUMERIC ASCII STRINGS (RECORDS)
RH      = 1             # OF BYTES IN RECORD HEADER

;
;   INTERNAL OPTION VALUES
;
ENSTOP  = 0             'EDGRUL' = STOP
ENWRAP  = 1             'EDGRUL' = WRAP SCREEN
ENREFL  = 2             'EDGRUL' = REFLECT OFF WALL
ENDIPR  = 3             'EDGRUL' = DISAPPEAR FROM SCREEN

MDRAW   = 0             'MODE' = DRAW
MDEBUG  = 1             'MODE' = DEBUG (ALL TEXT)
MSPLIT  = 2             'MODE' = SPLIT SCREEN WITH REGISTERS
MNORML  = 3             'MODE' = NORMAL -- SPLIT SCREEN W/O REGISTERS

SCRUNF  = 0             'SPEED' = RUN FULL SPEED
SCSTEP  = 1             'SPEED' = SINGLE STEP


;
;   COLLEEN DATA BASE
;
DOSVEC  = $000A         RESTART VECTOR LOCATION
APPMHI  = $000E         APPLICATION HIGH RAM
MEMLO   = $02E7         LOWEST AVAILABLE RAM POINTER
MEMHI   = $02E5         HIGHEST AVAILABLE RAM POINTER
BREAK   = $0011         BREAK KEY FLAG
LMARGN  = $0052         SCREEN LEFT MARGIN
RMARGN  = $0053         SCREEN RIGHT MARGIN
COLCRS  = $0055         SCREEN CURSOR COLUMN & (WORD)
ROWCRS  = $0054         SCREEN CURSOR ROW #
SPTCOL  = $0291         SPLIT SCREEN TEXT COLUMN (WORD)
SPTROW  = $0290         SPLIT SCREEN TEXT ROW
PCOLR0  = $02C0         PLAYER/MISSILE COLORS
PCOLR1  = $02C1
PCOLR2  = $02C2
PCOLR3  = $02C3
COLOR0  = $02C4         COLOR REGISTER 0
CH      = $02FC         KEYBOARD MATRIX CODE INPUT
CRSINH  = $02F0         CURSOR INHIBIT FLAG
DSPFLG  = $02FE         CONTROL BYTE DISPLAY FLAG
RTCLOK  = $0012         REAL-TIME CLOCK (FRAME COUNTER) 60HZ
DMACT   = $022F         DMA CONTROL BYTE
GPRIOR  = $026F         PLAYER/PLAYFIELD PRIORITY
PADDL0  = $0270         POT CONTROLLER 0 SENSE
STICK0  = $0278         JOYSTICK 0 SENSE
PTRIG0  = $D07C         POT 0 TRIGGER SENSE
STRIG0  = $0284         JOYSTICK 0 TRIGGER SENSE
;
;   HARDWARE EQUATES
;
HPOSP0  = $D000         PLAYER 0 HORIZONTAL POSITION
HPOSM0  = $D004         MISSILE 0 HORIZONTAL POSITION
HPOSM1  = $D005         MISSILE 1 HORIZONTAL POSITION
HPOSM2  = $D006         MISSILE 2 HORIZONTAL POSITION
HPOSM3  = $D007         MISSILE 3 HORIZONTAL POSITION
SIZEM   = $D00C         MISSILE SIZE CONTROL
GRAFM   = $D011         MISSILE [illegible]
DMACTL  = $D010         [illegible]
PMBASE  = $D407         PLAYER/MISSILE BASE ADDRESS REGISTER
[illegible]
```

```
AUDC1   = $D201         AUDIO #1 TYPE/VOLUME
RKVRND  = $D20A         POKEY RANDOM NUMBER
PACTL   = $D302         CASSETTE ON/OFF AMONG OTHER THINGS

GRAMON  = $01           'GRACTL' MISSILE DMA ON
DMACON  = $04           'DMACT' MISSILE DMA ON
;
;    ERROR MESSAGE EQUATES
;
ECSTKO  = 'S            STACK OVERFLOW
ECNEST  = 'N            NESTING ERROR -- UNMATCHED RIGHT BRACKET
ECDEFR  = 'R            DEFINE COMMAND USES RESERVED NAME
ECUDVF  = 'F            USER DEFINITION REGION FULL
ECINCL  = 'P            INCOMPLETE (PARTIAL) LINE INPUT
ECOLL   = 'O            OVERLENGTH INPUT LINE
ECABRT  = 'A            OPERATOR ABORT (BREAK KEY)
ECIOER  = 'I            SYSTEM I/O ERROR
ECUNDV  = 'U            UNDEFINED VARIABLE NAME USED
ECDNTL  = 'D            DEVICE NAME ERROR (TOO LONG)
ECOPEN  = 'I            GET/PUT DEVICE OPEN ERROR
ECLOAD  = 'L            LOAD ARGUMENT UNDEFINED
;
;   MISCELLANEOUS EQUATES
;
BUCKET  = $FFFFF        -1 INDICATES BIT BUCKET
```

```
;
;    RAM DATA BASE FOR TURTLE GRAPHICS PROGRAM
;
;        CONTROL REGION
;
;
=+$0080
EXEC    +=++1   0 = SCAN BUT DON'T EXECUTE, ELSE EXECUTE
KBIN    =+++1   0 = GET DATA FROM MEMORY, ELSE FROM KEYBOARD
;
;    START OF 'DTAB' REGION (USED FOR 'SYSXI' & 'DIXII' ROUTINES)
;
DTAB    = *
PTRBRH  *+++RH  RECORD INCLUDES THE 3 FOLLOWING POINTERS
        *+++RH
INPT    *+++3   INPUT LINE POINTER & OFFSET BYTE
        *++*RH
OUTPT   *+++3   OUTPUT LINE POINTER & OFFSET BYTE
        *++*RH
FLINE   *+++3   USER COMMAND LINE POINTER & OFFSET BYTE
;
        *++*RH
ACC     *+++NL  ARITHMETIC ACCUMULATOR
        *++*RH
NUMBER  *+++NL  INTERNAL NUMBER REGISTER
        *+++RH
LEVEL   *+++NL  USER COMMAND NESTING LEVEL
        *+++RH
CHAR    *+++1   CURRENT COMMAND BYTE
        *+++RH
ERR     *+++1   COMMAND ERROR CODE
SSTACK  *+++2   SOFTWARE STACK POINTER
XJUMP   *+++3   JUMP VECTOR
REDEF   *+++1   USER COMMAND REDEFINED FLAG
TEMP    *+++3   TEMPORARY WORK STORAGE FOR BUTTON LEVEL CODE SEQUENCER
COUNT   *+++1   WORK COUNTER
PTSTAT  *+*+1   'PTEST' TEMP
SWTEMP  *+++3   'SCNWRT' TEMP
KSTEMP  *+++1   '$SENSE' TEMP
AUDTMP  *+++2   '$AUDIO' TEMP
;
;    TURTLE GRAPHICS REGION
;
        *+++RH
XCURS   *+++2   X CURSOR (-32765 TO 32767)
YCURS   *+++2   Y CURSOR (-32768 TO 32767)
COLORN  *+++1   CURRENT COLOR # (NEGATIVE = PEN UP)
MODE    *+++1   CURRENT OPERATING MODE (0-3)
EDGRUL  *+++1   CURRENT EDGE RULE FOR COLLISIONS (0-3)
TRTREP  *+++1   CURRENT TURTLE REPRESENTATION (0-3)
AUDIO   *+++1   CURRENT AUDIO SELECT (0-15)
SPEED   *+++1   CURRENT TURTLE SPEED SELECTION (0-7)
NXTSCN  *+++1   NEXT SCREEN MODE (0-7)
SCNMOD  *+++1   CURRENT SCREEN MODE (0-7)
ORIENT  *+++1   CURRENT TURTLE ORIENTATION (0-7)
EEDGE   *+++1   EAST EDGE COLLISION SENSE
SEDGE   *+++1   SOUTH EDGE COLLISION SENSE
WEDGE   *+++1   WEST EDGE COLLISION SENSE
NEDGE   *+++1   NORTH EDGE COLLISION SENSE
TRTPOS  *+++1   TURTLE REPRESENTATION PLAYER POSITION (Y)
;
;    I/O DATA REGION
;
PDSPTB  *+++2   POINTER TO CURRENT DISPLAY TABLE
        *+++56
PROMPT  *+++1   USER PROMPT CHARACTER ('>' OR ' ')
INBUF   = *+0080+*+1    INPUT BUFFER
```

```
          *=*+80
LINBUF    *=*+INBUF

POEND     = *-1               *** MUST BE < $0100 ***
;
;     END OF 'DTAB' REGION
;
;     TURTLE PLAYER BUFFER
;
*=$0580             ORGED FOR MISSILES WITH BASE @ $0400

TPBUFF    *=*+128           MISSILES BUFFER AREA
TVBUFF    = TPBUFF+12       START OF VISIBLE REGION
TRBUFF    = TVBUFF-7        INCLUDES UNDERFLOW REGION
;
;     USER DEFINED VARIABLE REGION ('VDEF')
;
*=$0500

          *=*+RH
VDEF      *=*+VSIZE
          *=*+1    TERMINATOR BYTE

OPNBUF    *=*+DNSIZE+1      DEVICE NAME BUFFER FOR OPEN

P5END     = *-1             *** MUST BE < $0580 ***

;
;         UNUSED PAGE (FREE FOR ANY USE)
;
*=$0600

P6END     = *-1             *** MUST BE < $0700 ***
```

```
        IF      BOOT
        IF      DLOAD
*=$2000
        ENDIF
        IF      DLOAD-1
*=$0700
        ENDIF
        ENDIF
        IF      BOOT-1
*=$A800
        ENDIF

; CASSETTE BOOT FILE INFORMATION

        IF      BOOT
        IF      DLOAD-1
PST=*
        BYTE    0               (DOESN'T MATTER)
        BYTE    PND-PST+127/128 NUMBER OF RECORDS
        WORD    PST             MEMORY ADDRESS TO START LOAD
        WORD    PINIT           PROGRAM INIT VECTOR

; ENTRY POINT FOR MULTI-STAGE BOOT PROCESS

        CLC                     SET PROPER "NO ERROR" STATUS
        RTS

; ENTRY POINT FOR FIRST TIME INITIALIZATION

PINIT   LDA     #$3C            TURN OFF CASSETTE
        STA     PACTL

        ENDIF

        LDA     #PND            ESTABLISH UPPER MEMORY LIMIT
        STA     MEMLO
        LDA     #PND/256
        STA     MEMLO+1

        LDA     #RESTRT         ESTABLISH JUMP VECTOR
        STA     DOSVEC
        LDA     #RESTRT/256
        STA     DOSVEC+1

        IF      DLOAD-1
        RTS
        ENDIF

        ENDIF

;
; TURTLE INITIALIZATION
;

; POWER-UP ENTRY POINT

        IF      BOOT-1
INIT    RTS                     RETURN TO POWER-UP ROUTINE.
        ENDIF

; WARMSTART ENTRY POINT (RESET KEY)

RESTRT  LDX     #$FF            SETUP HARDWARE STACK POINTER
        TXS

        LDA     #$0C            CLEAR UPPER [illegible]
```

```
          LDA   #0
INIOIO    STA   0,X
          INX
          CPX   #0              PAGE WRAP POINT?
          BNE   INIOIO          NO -- CONTINUE

          LDA   #$4C            PUT "JMP" OPCODE IN JUMP VECTOR
          STA   JUMPAD

          LDA   #TWHAT          SETUP MESSAGE TABLE POINTER
          STA   POSPTR
          LDA   #TWHAT/256
          STA   POSPTR+1

          LDX   #IOCB0          INIT IOCBS 0,1,2 & 3

INIO2O    LDA   #0
          STA   ICAUX1,X
          STA   ICAUX2,X
          STA   ICBLL,X
          STA   ICBLH,X
          LDA   #OPNBUF
          STA   ICBAL,X
          LDA   #OPNBUF/256
          STA   ICBAH,X
          TXA
          CLC
          ADC   #IOCBRT         BUMP TO NEXT IOCB
          TAX
          CPX   #IOCB#          DONE?
          BNE   INIO2O          NO -- DO NEXT IOCB

          LDA   #1              SET (NON-ZERO) OPTION DEFAULT VALUES
          STA   COLORN
          LDA   #6              HARDWARE SCREEN MODE #7
          STA   NXTSCR
          STA   SCNMOD
          LDA   #PNORML
          STA   MODE
          LDA   #ERDIPR         EDGE RULE IS "REFLECT"
          STA   EDGRUL
          LDA   #5              TURTLE REPRESENTATION IS "ON"
          STA   TRTREP

          LDX   #128            CLEAR TURTLE REPRESENTATION BUFFER (PLAYER)
          LDA   #0

INI022    STA   TPBUFF-1,X
          DEX
          BNE   INI022

          LDX   #8              INITIALIZE PLAYER/MISSILE HARDWARE
          LDA   #0

INI027    STA   HPOSO,X         SET ALL HORIZONTAL POSITIONS TO ZERO
          DEX
          BNE   INI027

          LDA   #$0C                PRIORITY
          STA   GPRIOR          (GLOBAL RAM)
          LDA   #TPBUFF-$180/256    PLAYER BASE ADDRESS REGISTER
          STA   PMBASE
          LDA   #$0C                PLAYER SIZE
          STA   SIZEA

          JSR   MODSEL          OPEN ALL IOCBS
```

```
        STA     PAGE
        LDA     #ACC+           PLAYER SIZE
        STA     SIZE
;
        JSR     MODSEL          OPEN ALL IOCBS
;
        JSR     $HOME+3         HOME CURSOR (TURTLE)
        JSR     $NORTH+3        FACE TURTLE NORTH
        JSR     $CLEAR+3        CLEAR SCREEN
;
        JSR     PLOTRT          PLACE TURTLE REPRESENTATION
;
        LDA     #1              SETUP ALL RECORD LENGTHS
        STA     CHAR+1
        STA     PROMPT+1
        STA     ERR+1
        LDA     #3
        STA     INPT+1
        STA     OUTPT+1
        STA     FLINE+1
        LDA     #4
        STA     XCURS+1
        LDA     #NL
        STA     LEVEL+1
        STA     NUMBER+1
        STA     ACC+1
        LDA     #INSIZ
        STA     LININ+1
        LDA     #VSIZE
        STA     VDEF+1
        LDA     #3+RH+1
        STA     PTRHRH
;
        LDX     #ACC-START      ZERO 'ACC'
        JSR     SCLR1
;
        JSR     CLRUDF          CLEAR USER DEFINED COMMANDS
        JSR     CLRVDF              AND VARIABLES
;
        LDA     #$FF            RESET BREAK KEY FLAG
        STA     BREAK
;
        JSR     CLRINL          CLEAR INPUT LINE BUFFER
;
        LDA     #'              (BLANK)
        STA     CHAR            BLANK TO 'CHAR'
        JMP     DIRECT              & 'ERR'
```

```
; THIS IS THE MAIN-LINE LOOP --- STACKS ARE INITIALIZED, DELETED
; USER COMMANDS ARE CLEARED, AND A SINGLE COMMAND IS EXECUTED
; 'DIRECT' IS THE EXTERNAL ENTRY POINT FOR ABORT AND FATAL ERROR CONDITIONS
;
DIRECT  LDX     #$FF            RE-INIT HARDWARE STACK POINTER
        TXS
        STA     ERR             SAVE "ERROR" CODE
        LDA     MEMLO           INIT SOFTWARE STACK POINTER
        CLC
        ADC     #USIZE+1        STARTS AFTER 'UDEF' AREA.
        STA     SSTACK
        LDA     MEMLO+1
        ADC     #USIZE+1/256
        STA     SSTACK+1
        LDX     #LEVEL-DTAB     SET USER COMMAND LEVEL TO 0
        JSR     SCLR1
        LDX     #NUMBER-DTAB    SET 'NUMBER' TO ZERO
        JSR     SCLR1
ML**                            MAIN-LINE LOOP
        LDA     REDEF           USER COMMAND REDEFINED?
        BEQ     ML10            NO
        LDA     #0              YES -- RE-ALLOCATE MEMORY USED
        STA     REDEF
ML05    LDA     #DEL            FIND DELETED COMMANDS
        STA     CHAR
        JSR     UFIND
        BNE     ML10            NO MORE FOUND
        LDY     #0
        LDA     #EOL            EOL DEALLOCATES THE BUFFER AREA
        STA     (FLINE),Y
        JMP     ML05
ML10    JSR     COMMND          EXECUTE ONE COMMAND
        JMP     ML
```

```
; THIS IS THE COMMAND INITIATOR -- A COMMAND IS READ FROM THE KEYBOARD
; AND STORED IN THE WORK BUFFER (=LININ) AS IT IS SYNTAX CHECKED, AND
; THEN EXECUTED.  SCREEN REDIRECTING IS HANDLED HERE ALSO
;
COMMAND LDA     #':             SETUP INPUT PROMPT
        STA     PROMPT
;
        JSR     XWHAT           PUT INFO TO SCREEN
        JSR     CLRINL              THEN CLEAR INPUT LINE BUFFER
;
        LDA     #$FF            GET INPUT FROM KEYBOARD:
        STA     #BIN
        LDA     #0                  SCAN WITHOUT EXECUTING
        STA     EXEC
        LDA     #LININ              & STORE TO LINE BUFFER
        STA     OUTPT
        LDA     #LININ/256
        STA     OUTPT+1
        LDA     #0
        STA     OUTPT+2
;
        JSR     RCMD            GET (SCAN) ONE COMMAND
;
        JSR     SCNEOL          SCAN TO "EOL" IF INPUT FROM "E:"
;
        INC     #BIN            (=0) GET INPUT FROM LINE BUFFER
        LDA     #LININ
        STA     INPT
        LDA     #LININ/256
        STA     INPT+1
        LDA     #0
        STA     INPT+2
;
        STA     OUTPT+2             & STORE TO BIT BUCKET
        LDA     #BUCKET
        STA     OUTPT
        LDA     #BUCKET/256
        STA     OUTPT+1
        DEC     EXEC            (=-1)     WHILE EXECUTING INPUT LINE
;
        LDA     #               (BLANK) -- REMOVE PROMPT WHILE EXECUTING.
        STA     PROMPT
        STA     ERR             ALSO REMOVE ERROR CODE FROM PRIOR LINE.
;
        JSR     XWHAT           SETUP REPORT TO SCREEN.
;
        JSR     RCMD            EXECUTE ONE COMMAND.
;
        BIG                     *** CHANGE ABOVE TO JMP WHEN DEBUGGED ***
```

```
; SYNTAX SCANNER AND EXECUTER -- IF 'EXEC' > 0, THEN SCAN ONE COMMAND AND RETURN.
; IF 'EXEC' <> 0, THEN ONE COMMAND IS EXECUTED.  'CMD' IS AN EXTERNAL
; ENTRY POINT THAT ASSUMES THAT A COMMAND IS IN 'CHAR'.
; COMMANDS ARE EITHER INTRINSIC (PART OF SYSTEM) OR USER DEFINED.
;
RCMD    JSR     GETCH           GET CHARACTER.

; *** EXTERNAL ENTRY POINT ***

CMD     JSR     WRITEV          CHECK FOR BREAK KEY.
        JSR     PRICOM          CHECK FOR PRIORITY COMMAND PENDING.
        JSR     RUNOPT          PROCESS RUN OPTIONS.
        LDA     CHAR            EXPLICIT USER COMMAND INVOCATION?
        CMP     #'*
        BEQ     CMD015          YES -- IGNORE RESERVED WORD CHECK.
        JSR     RESERV          NO -- CHECK FOR INTRINSIC COMMAND.
        BNE     CMD020          NOT INTRINSIC -- SEE IF USER COMMAND.
;       INTRINSIC COMMAND.
        LDA     EXEC            YES -- SET CC FOR EXEC/SCAN OPTION.
        JSR     XJUMP           EXECUTE X-ROUTINE.
        RTS                     *** CHANGE ABOVE TO JMP WHEN DEBUGGED ***
;       NOT INTRINSIC COMMAND.
CMD015  JSR     GETCH           GET USER COMMAND NAME (IGNORE *).
CMD020  LDA     EXEC            EXECUTE?
        BNE     CMD023          YES.
CMD021  RTS                     NO.
CMD023  JSR     UFIND           USER DEFINED COMMAND?
        BNE     CMD021          NO -- IGNORE (SLOW NOP).
;       USER DEFINED COMMAND.
        LDX     #INPT-DTAB      YES -- SAVE INPUT SCAN POINTER.
        JSR     SPSHI           PUSH POINTER TO STACK.
        TSX                     SEE IF STACK FULL ENOUGH TO WORRY ABOUT.
        CPX     #$80
        BCS     CMD032          NO.
        JSR     PUSHHS          YES -- PUSH HARDWARE STACK TO SOFTWARE STACK.
CMD032  LDA     FLINE           ESTABLISH USER COMMAND AS NEW INPUT SCAN LINE.
        STA     INPT
        LDA     FLINE+1
        STA     INPT+1
        LDA     #2              SKIP OVER "X="
        STA     INPT+2
        LDX     #LEVEL-DTAB     INCREMENT USER LEVEL #.
        JSR     SINCT
;       EXECUTE USER COMMAND.
        JSR     RCMD            PROCESS USER-DEFINITION.
        LDX     #LEVEL-DTAB     DECREMENT USER LEVEL #.
```

```
            JSR     SINCI

;   EXECUTE USER COMMAND

            JSR     RCMD            PROCESS USER-DEFINITION

            LDX     #LEVEL-DTAB     DECREMENT USER LEVEL #
            JSR     SDCRI

            TSX                     HARDWARE STACK EMPTY?
            CPX     #$FF
            BNE     CMD040          NO

            JSR     PULLHB          YES -- PULL DATA FROM SOFTWARE STACK.

CMD040      LDX     #INPT-DTAB      RE-ESTABLISH INPUT SCAN LINE
            JSR     SPULI           PULL POINTER FROM STACK
            RTS                     *** DON'T CHANGE ABOVE JSR TO JMP !!! ***
```

```
;  RESERV -- SEARCH CHARACTER FOR SYSTEM INTRINSIC
;
;  CALLING SEQUENCE
;
;       'CHAR' = CHARACTER IN QUESTION
;       'CTAB' = COMMAND JUMP TABLE
;
;       JSR     RESERV
;       BNE     NOT RESERVED INTRINSIC
;
;       'XJUMP' = JUMP TO X-ROUTINE IF FOUND
;
RESERV  LDA     CHAR            GET COMMAND NAME
        SEC                     (CLEAR BORROW)
        SBC     #$20            NORMALIZE BLANK TO 00
        CMP     #$60            IN INTRINSIC SPACE?
        BCC     RES010          YES -- COULD BE INTRINSIC
;
RES005  LDA     #$FF            NO -- NOT RESERVED WORD
        RTS
;
RES010  ASL     A               X2 FOR ACCESS TO ADDRESS TABLE
        TAX
        LDA     CTAB+1,X        GET MSB OF ADDRESS
        BEQ     RES005          NO ENTRY -- NOT RESERVED
;
        STA     XJUMP+2
        LDA     CTAB+0,X        GET LSB OF ADDRESS
        STA     XJUMP+1
        LDA     #0              SET CC FOR EXIT
        RTS
```

```
; COMMAND TABLE
;
; EACH ENTRY (ORDERED IN ATASCII SEQUENCE) IF THE ADDRESS
; OF THE COMMAND PROCESSOR ROUTINE OR ZERO
;
CTAB==*
        WORD    XNOP            BLANK = NOP
        WORD    XSTOP           ! = STOP ITERATION (ONE LEVEL)
        WORD    0               "
        WORD    XVAR            # = ITERATE BY VARIABLE
        WORD    XJOYS           $ = JOYSTICK TEST
        WORD    XPOT            % = READ POT CONTROLLER TO ACCUMULATOR
        WORD    XCOLOR          & = COLOR REGISTER UPDATE
        WORD    0               '
        WORD    XLPARN          ( = NESTING BRACKET
        WORD    XNERR           ) = ILLEGAL WITHOUT (
        WORD    0               * = RESERVED FOR COMMAND DELIMITER
        WORD    XPLUS           + = INCREMENT ACCUMULATOR
        WORD    0               ,
        WORD    XMINUS          - = DECREMENT ACCUMULATOR
        WORD    0               .
        WORD    0               /
        WORD    XITER           0 = ITERATE
        WORD    XITER           1 = ITERATE
        WORD    XITER           2 = ITERATE
        WORD    XITER           3 = ITERATE
        WORD    XITER           4 = ITERATE
        WORD    XITER           5 = ITERATE
        WORD    XITER           6 = ITERATE
        WORD    XITER           7 = ITERATE
        WORD    XITER           8 = ITERATE
        WORD    XITER           9 = ITERATE
        WORD    0               :
        WORD    XCMPAB          ; = DIRECTION SENSE
        WORD    0               <
        WORD    XDEFIN          = = DEFINE USER COMMAND OR VARIABLE
        WORD    0               >
        WORD    XQUEST          ? = RANDOM TEST

        WORD    XZERO           @ = SET ACC TO ZERO
        WORD    XA              A = ITERATE BY ACCUMULATOR
        WORD    XBEEP           B = BEEP
        WORD    XCLEAR          C = CLEAR SCREEN
        WORD    XDOWN           D = PEN DOWN
        WORD    XEDGE           E = EDGE TEST
        WORD    XFRWRD          F = TURTLE FORWARD
        WORD    0               G
        WORD    XHOME           H = TURTLE HOME
        WORD    0               I
        WORD    0               J
        WORD    0               K
        WORD    XROTL           L = ROTATE TURTLE LEFT
        WORD    0               M
        WORD    XNORTH          N = FACE TURTLE NORTH
        WORD    0               O
        WORD    XPEN            P = PEN COLOR SELECT
        WORD    0               Q
        WORD    XROTR           R = ROTATE TURTLE RIGHT
        WORD    XSENSE          S = TURTLE COLOR SENSE
        WORD    XIF             T = ACCUMULATOR TEST
        WORD    XUP             U = PEN UP
        WORD    0               V
        WORD    XWAIT           W = WAIT FOR NEXT CLOCK TICK
        WORD    0               X
```

```
WORD    0               V
WORD    0               I
WORD    XPUSHA          C = NESTING BRACKET WITH ACC PUSH
WORD    0               \
WORD    XNERR           ] = ILLEGAL WITHOUT [
WORD    XBUMP           ^ = BUMP ITERATION COUNT
WORD    XNOP            (UNDERSCORE) = NOP

WORD    0
WORD    XAUDIO          A = SELECT AUDIO INTPUT
WORD    0               B
WORD    XCLRV           C = CLEAR USER VARIABLES
WORD    XDSPMD          D = SELECT DISPLAY MODE
WORD    XERULE          E = SELECT EDGE RULE
WORD    0               F
WORD    XGETFL          G = GET USER DEFS
WORD    0               H
WORD    0               I
WORD    0               J
WORD    0               K
WORD    XLOAD           L = LOAD CANNED PROGRAMS
WORD    XMODE           M = SELECT OPERATING MODE
WORD    0               N
WORD    0               O
WORD    XPUTFL          P = PUT USER DEFS
WORD    0               Q
WORD    XRUN            R = LOAD & RUN CANNED PROGRAMS
WORD    XSPEED          S = SELECT SPEED
WORD    XTREP           T = SELECT TURTLE REPRESENTATION
WORD    0               U
WORD    0               V
WORD    0               W
WORD    0               X
WORD    0               Y
WORD    XRESET          Z = SOFT RESET
WORD    0
WORD    0
WORD    0
WORD    0
WORD    0
```

```
; UFIND -- FIND USER COMMAND DEFINITION IF PRESENT
;
; CALLING SEQUENCE:
;
;       'CHAR' = COMMAND NAME
;       'UDEF' AREA STARTS AT BOTTOM OF AVAILABLE MEMORY
;
;       JSR     UFIND
;       BNE     NOT FOUND
;
;       'FLINE' = POINTER TO COMMAND DEFINITION, IF FOUND
;
;       REGISTER X IS CLOBBERED, Y = 0

UFIND   JSR     SETUDF          SETUP POINTER TO 'UDEF' AREA
        LDA     #LINSIZ
        STA     FLINE+2

; *** EXTERNAL ENTRY POINT ***

XFIND   LDY     #0              SEARCH 1ST CHAR OF EACH LINE
        LDA     (FLINE),Y       GET CHARACTER
        CMP     #$FF            END OF TABLE INDICATOR?
        BNE     FND020          NO -- CHECK FOR MATCH

        LDA     #$FF            YES -- SET CC FOR EXIT

FND010  RTS                     RETURN WITH CC SET

FND020  CMP     CHAR            IS THIS THE ONE WE'RE LOOKING FOR?
        BEQ     FND010          YES -- RETURN WITH CC SET

        LDX     #FLINE-DTAB     NO -- TRY AGAIN.
        LDY     FLINE+2         GET INCREMENT TO NEXT DEFINITION.
        JSR     FADDY           INCREMENT 'FLINE'
        JMP     XFIND

; VFIND -- FIND USER DEFINED VARIABLE, IF PRESENT
;
; CALLING SEQUENCE:
;
;       'CHAR' = NAME OF VARIABLE
;       'VDEF' AREA HAS VARIABLE VALUES
;
;       JSR     VFIND
;       BNE     NOT FOUND
;
;       'FLINE' = POINTER TO VARIABLE DEFINTION, IF FOUND
;
;       REGISTER X IS CLOBBERED, Y = 0

VFIND   LDA     #VDEF           SETUP POINTER TO DEFINITIONS
        STA     FLINE
        LDA     #VDEF/256
        STA     FLINE+1
        LDA     #VLNGTH         LENGTH OF EACH DEFINITION
        STA     FLINE+2
        JMP     XFIND           GO TO COMMON CODE ('UFIND' & 'VFIND')
```

```
; RUNOPT -- PROCESS RUN-TIME OPTIONS
;
RUNOPT  LDA     EXES            CHECK FOR EXECUTE OR SCAN
        BEQ     RUN010          SCAN -- NO OPTION CONTROL
;
        LDA     MODE            CHECK MODE
        CMP     #MGRAM          FULL-GRAPHICS?
        BEQ     RUN010          YES -- NO TEXT TO SCREEN
;
        CMP     #MNORML         NORMAL MODE?
        BEQ     RUN010          YES -- NO TEXT TO SCREEN
;
        LDY     #TVARS-TWHAT    NO -- PUT VARIABLES TO SCREEN
        JSR     SCRWRT
;
;       PROCESS SPEED OPTION
;
RUN010  LDA     SPEED           CHECK OPTION
        BEQ     RUN050          0 = FULL SPEED AHEAD
;
        CMP     #SCSTEP         SINGLE STEP?
        BEQ     RUN030          YES
;
        SEC                     (CLEAR BORROW)
        SBC     #1              NO -- SYNCHRONIZE SPEED TO CLOCK
        JSR     CLKSYN
        JMP     RUN050
;
;       SINGLE STEP -- WAIT FOR ANY KEY STROKE
;
RUN030  LDA     #$FF            KEYSTROKE?
        CMP     CH
        BNE     RUN033          YES
;
        JSR     ABRTCK          BREAK KEY?
        JMP     RUN030          NO -- WAIT FOR ONE OR THE OTHER
;
RUN033  STA     CH              RESET KEYSTROKE
;
; SET AUDIO IF SELECTED
;
RUN050  LDA     AUDIO           AUDIO SELECTED?
        BEQ     RUN090          NO
;
        TSX
        STX     AUDTMP+2        (FOR SOUND GENERATION)
        ASL     A
        TAX
        LDA     AUDTAB,X        GET L.S.B. OF ADDRESS
        STA     AUDTMP
        LDA     AUDTAB+1,X      GET M.S.B. OF ADDRESS
        STA     AUDTMP+1
        LDY     #0
        LDA     (AUDTMP),Y      GET VALUE AT THE ADDRESS
        AND     #$FF
        CLC
        ADC     #$00
        STA     AUDF1           FREQUENCY SELECT
        LDA     #$A0+8
        STA     AUDC1
;
RUN090  RTS
;
AUDTAB  =*-2
```

```
RUN090  RTS

AUDTAB  = *-2
        .WORD   COLCRS          LSB     1       A
        .WORD   ROWCRS          LSB     2       B
        .WORD   SSTACK+1        MSB     3       C
        .WORD   PADDLO          LSB     4       D
        .WORD   PADDLO+1        LSB     5       E
        .WORD   CHAR                    6       F
        .WORD   TEMP                    7       G
        .WORD   TEMP+1                  8       H
        .WORD   COUNT                   9       I
        .WORD   AUDTMP+2        S       10      J
        .WORD   XCURS+1         MSB     11      K
        .WORD   YCURS+1         MSB     12      L
        .WORD   SSTACK          LSB     13      M
        .WORD   INPT+2                  14      N
        .WORD   ACC+NL-1        LSB     15      O
```

```
;
; CLRINL -- CLEAR INPUT LINE UTILITY
;
; CALLING SEQUENCE:
;
;       JSR     CLRINL
;
;       'LININ' IS SET TO BLANKS WITH AN EOL AT THE END
;
CLRINL  LDX     #INSIZ          CLEAR INPUT LINE
        LDA     #' '            (BLANK)
;
CILO10  STA     LININ-1,X
        DEX
        BNE     CILO10
;
        LDA     #EOL            TERMINATE LINE FOR PRINTING.
        STA     LININ+INSIZ-1
        RTS
;
;
; SCNEOL -- SCAN TO EOL CHARACTER IF INPUT FROM "E:" DEVICE
;
; CALLING SEQUENCE:
;
;       JSR     SCNEOL
;
SCNEOL  LDX     MODE            INPUT FROM "E:"?
        LDA     TINX,X
        CMP     #4
        BNE     SCE090          NO
;
SCE010  JSR     CHIN            YES -- SCAN
        CMP     #EOL
        BNE     SCE010          ... TO EOL CHARACTER.
;
SCE090  RTS
```

```
;*** START OF LEVEL 4 OVERLAY(4) --- COMMAND PROCESSOR (E-STATION) ***

;  USER DEFINED COMMAND PROCESSOR --- =CHAR=:COMMAND
;                                     ==WORD=:COMMAND
;
XDSTIN  JSR     GETCH           GET COMMAND/VARIABLE NAME
        CMP     #'&             IS IT A USER VARIABLE DEFINITION?
        BEQ     XDF100          YES

        CMP     #'=             NO -- IS IT THE RESERVED WORD OVERRIDE CHAR?
        BEQ     XDF005          YES

        JSR     RESERV          IS IT RESERVED (INTRINSIC)?
        BNE     XDF010          NO

        LDA     EXEC
        BEQ     XDF015          NON-EXECUTE -- DEFER ERROR REPORTING

        LDA     #ECDEFP
        JMP     DIRECT          YES -- FATAL ERROR

;    USER COMMAND DEFINITION --- =NAME=:COMMAND

XDF005  JSR     GETCH           GET COMMAND NAME (IGNORE =)

XDF010  LDA     EXEC            EXECUTE MODE?
        BNE     XDF020          YES

XDF015  JMP     RCMD            NO -- SCAN DEFINITION & RETURN

XDF020  JSR     UFIND           USER COMMAND ALREADY DEFINED?
        BNE     XDF025          NO

        LDA     #DEL            YES -- MARK IT AS DELETED NOW
        STA     (FLINE),Y
        STA     REDEF           SET FLAG FOR SPACE RECLAMATION LATER.

XDF025  LDA     CHAR            SAVE 'CHAR' (COMMAND NAME).
        PHA
        LDA     #EOL            NOW SCAN FOR BLANK ENTRY FOR NEW DEF.
        STA     CHAR
        JSR     UFIND
        BEQ     XDF030          GOOD -- FOUND A SPOT

        LDA     #EQUOVF
        JMP     DIRECT          NO ROOM FOR DEFINITION -- ABORT

XDF030  JSR     GETCH           GET 1ST CHARACTER OF DEFINITION
        CMP     #' '            IS IT BLANK (MEANS DELETE)?
        BEQ     XDF050          YES -- NO MORE TO DO

        LDY     #0              NO -- ENTER NEW DEFINITION.
        PLA                     GET AND STORE COMMAND NAME.
        STA     (FLINE),Y
        INY
        LDA     #'=             STORE '='
        STA     (FLINE),Y
        INY
        LDA     CHAR            STORE 1ST BYTE OF DEFINITION.
        STA     (FLINE),Y

        LDX     #OUTPTR+XTAL    SAVE OUTPUT POINTER
```

```
         JSR     SPSHT           PUSH TO STACK

         LDA     FLINE           ROUTE OUTPUT TO 'FLINE' (UDEF AREA)
         STA     OUTPT
         LDA     FLINE+1
         STA     OUTPT+1
         LDA     #3              SKIP OVER "X=X" ALREADY STORED
         STA     OUTPT+2
         INC     EXEC            (=0) GET TO SCAN MODE (NO EXECUTE)

         JSR     CMD             COPY DEFINTION TO BUFFER W/O EXECUTE.

         LDY     OUTPT+2         NOW ADD EOL AT END OF DEFINITION.
         LDA     #EOL
         STA     (OUTPT),Y

         DEC     EXEC            (=-1) SET BACK TO EXECUTE MODE.
         LDX     #OUTPT-DTAB     RESTORE OUTPUT POINTER.
         JSR     SPULI
         RTS                     *** DON'T CHANGE ABOVE JSR TO JMP ***

XDF05C   PLA                     CLEAR STACK BEFORE LEAVING.
         RTS
```

```
;    VARIABLE ASSIGNMENT TO ACC CONTENTS -- =#VAR
;
XDF100  JSR     GFLD            GET VARIABLE NAME
        LDA     CMDEC           EXECUTE MODE?
        BEQ     XDF190          NO
;
        JSR     VFIND           ALREADY DEFINED?
        BEQ     XDF135          YES -- ASSIGN NEW VALUE
;
        LDA     CHAR            NO -- LOOK FOR FREE SPOT
        PHA                     SAVE 'CHAR'.
        LDA     #' '            SEARCH FOR BLANK
        STA     CHAR
        JSR     VFIND           FIND EMPTY SPOT?
        BEQ     XDF130          YES
;
        LDA     #ECUOVF
        JMP     DIRECT          NO -- ABORT
;
XDF130  PLA                     RESTORE 'CHAR'
        STA     CHAR
        LDY     #0              SAVE IT AS VARIABLE NAME
        STA     (FLINE),Y
        INY
        LDA     #'=             SAVE "="
        STA     (FLINE),Y
;
XDF135  LDX     #0              NOW SAVE CURRENT 'ACC' CONTENT AS VALUE
        LDY     #2
;
XDF140  LDA     ACC,X           GET DIGIT
        STA     (FLINE),Y       SAVE IN DEFINITION.
        INX
        INY
        CPX     ACC+1           DONE YET?
        BNE     XDF140          NO.
;
XDF190  RTS                     YES.
;
;
;    CLEAR USER DEFINED VARIABLE REGION -- CTRL-C
;
XCLRV   BEQ     XCV090          NON-EXECUTE
;
        JSR     CLRVDF          CLEAR VARIABLE REGION
;
XCV090  RTS
```

```
;
; COMMAND PROCESSOR FOR ACCUMULATOR NON-ZERO TEST -- ?<THEN><ELSE>
;
; 'SKIP' & 'TEST' ARE EXTERNAL ENTRY POINTS USED BY THE
;  EDGE TEST AND RANDOM TEST ROUTINES ALSO
;
XIF     BNE     XIFD10          EXECUTE
;
;     *** EXTERNAL ENTRY POINT ***
;
SKIP    JSR     RCMD            SCAN COMMAND (THEN).
        JMP     RCMD            SCAN COMMAND (ELSE) & RETURN.
;
XIFD10  LDX     #ACC-DTAB       SEE IF 'ACC' = ZERO
        JSR     TSTNUM
;
;     *** EXTERNAL ENTRY POINT ***
;
TEST    BNE     XIFD20          NO -- EXECUTE "THEN" COMMAND
;
;    EXECUTE ELSE
;
        INC     EXEC            (=0) SCAN 1ST COMMAND (THEN)
        JSR     RCMD
        DEC     EXEC            (=-1) EXECUTE 2ND COMMAND (ELSE)
        JMP     RCMD              & RETURN.
;
;    EXECUTE THEN
;
XIFD20  JSR     RCMD            EXECUTE 1ST COMMAND (THEN).
        INC     EXEC            (=0) SCAN 2ND COMMAND (ELSE)
        JSR     RCMD
        DEC     EXEC            (=-1) RESTORE EXECUTE MODE.
        RTS
;
;
; RANDOM TEST COMMAND PROCESSOR -- ?<THEN><ELSE>
;
XQUEST  BNE     XQUD10          EXECUTE MODE.
;
        JMP     SKIP            SCAN MODE -- SCAN BOTH THEN & ELSE COMMANDS.
;
XQUD10  LDA     RKVRND          GET RANDOM NUMBER FROM POKEY CHIP
        AND     #1              MASK DOWN TO BINARY DECISION (SET CC).
        JMP     TEST            NOW PROCESS THEN OR ELSE BASED ON RESULT.
```

```
;
; COMMAND PROCESSOR FOR NESTING OPERATOR -- (<COMMAND>  <COMMAND>)
;
XLP010  JSR     CMD             PROCESS COMMAND.

XLPARN  JSR     GETCH           GET NEXT COMMAND (OR CLOSING PAREN).
        CMP     #')             CLOSING PAREN?
        BNE     XLP010          NO -- PROCESS COMMAND.

        JSR     RUNOPT          TREAT ")" AS A COMMAND.

XNOP    RTS                     YES -- NESTING COMPLETE.


;
; COMMAND PROCESSOR FOR PUSH/POP OPERATORS -- [<COMMAND>  ,  <COMMAND>]
;
XPUSHA  BEQ     XPA010          NON-EXECUTE.

        LDX     #ACC-DTAB       EXECUTE -- PUSH ACCUMULATOR.
        JSR     SPSHI

XPA010  JSR     GETCH           GET NEXT COMMAND (OR MATCHING BRACKET).
        CMP     #']             MATCHING BRACKET?
        BEQ     XPA020          YES.

        JSR     CMD             NO -- PROCESS COMMAND.
        JMP     XPA010

XPA020  JSR     RUNOPT          TREAT "]" AS COMMAND.
        LDA     EXEC
        BEQ     XPA090          NON-EXECUTE.

        LDX     #ACC-DTAB       EXECUTE -- PULL ACCUMULATOR.
        JSR     SPULI

XPA090  RTS                     *** DON'T CHANGE ABOVE JSR TO JMP !!! ***


XNERR   JSR     SCNEOL          SCAN TO EOL IF INPUT FROM "E "

        LDA     #ECNEST         NESTING ERROR -- UNMATCHED RIGHT BRACKET.
        JMP     DIRECT
```

```
; ACCUMULATOR ITERATE COMMAND PROCESSOR -- ACCUMULATOR
;
CR      BNE     WRITE           EXECUTE
        JMP     SCMD            SCAN COMMAND & RETURN
;
WRITE   LDI     #NUMBER-DTAB    SAVE CURRENT 'NUMBER' VALUE
        JSR     SPSHI
;
        LDI     #ACC-DTAB       MOVE 'ACC' TO 'NUMBER'
        LDY     #NUMBER-DTAB
        JSR     SMOVI
;
        JSR     SCTCH           GET NEXT COMMAND
;
        JMP     ITERB           GO TO COMMON CODE TO ITERATE COMMAND
;
;
; NUMBER ITERATE COMMAND PROCESSOR -- (NUMBER=COMMAND)
;
XITER   LDX     #NUMBER-DTAB    SAVE CURRENT 'NUMBER' VALUE
        JSR     SPSHI
;
        JSR     NUMB            GET NEW VALUE TO 'NUMBER' (SCANS TO NEXT COMMAND)
        LDA     EXEC            EXECUTE MODE?
        BNE     ITERB           YES -- GO TO COMMON CODE FOR ITERATION
;
        JSR     CMD             NO -- SKIP NEXT COMMAND
        JMP     XITOPO              & RESTORE 'NUMBER'
;
;       *** EXTERNAL ENTRY POINT ***
;
ITERB   LDA     CHAR            SEE IF ASSIGNMENT OPERATOR.
        CMP     #'@
        BNE     XITOSO          NO -- ITERATION
;
        LDX     #NUMBER-DTAB    YES -- MOVE ITERATION COUNT TO ACC.
        LDY     #ACC-DTAB
        JSR     SMOVI
        JMP     XITO60              & DON'T ITERATE.
;
XITOSO  LDX     #NUMBER-DTAB    COMMON CODE -- TEST NUMBER
        JSR     TSTNUM
        BEQ     XITO60          = ZERO, DONE
;
        LDX     #INPT-DTAB      SAVE INPUT POINTER INFORMATION
        JSR     SPSHI
        LDA     CHAR            SAVE 'CHAR'
        PHA
;
        JSR     CMD             EXECUTE COMMAND.
;
        PLA                     RESTORE 'CHAR'
        STA     CHAR
        LDX     #INPT-DTAB      RESTORE INPUT POINTER
        JSR     SPULI
;
        LDX     #NUMBER-DTAB    DECREMENT 'NUMBER'
        JSR     SDCRI
        JMP     ITERB           CHECK FOR ANOTHER ITERATION
;
XITO60  INC     EXEC            (=0) ALL DONE -- SCAN OVER COMMAND
        JSR     CMD
        DEC     EXEC            (=-1) RESTORE EXECUTE MODE
```

```
                JSR     $DCRI           DECREMENT 'NUMBER'
                JMP     ITERR0          CHECK FOR ANOTHER ITERATION

XITOA0  INC     EXEC            (=0) ALL DONE -- SCAN OVER COMMAND
                JSR     CMB
                DEC     EXEC            (=-1) RESTORE EXECUTE MODE

XITO70  LDX     #NUMBER-DTAB    RESTORE ORIGINAL 'NUMBER' VALUE
                JSR     $PULI           (& CLEAN STACK).
                RTS                     *** DON'T CHANGE ABOVE JSR TO JMP (!) ***


;
; STOP ITERATION (INNER LEVEL) COMMAND PROCESSOR -- (
;
XSTOP   BEQ     XST090          NON-EXECUTE

        LDX     #NUMBER-DTAB    CLEAR 'NUMBER' TO STOP ITERATION
        JSR     $CLRI

XST090  RTS


;
; BUMP ITERATION COUNT (INNER LEVEL) COMMAND PROCESSOR -- ^
;
XBUMP   BEQ     XBM090          NON-EXECUTE

        LDX     #NUMBER-DTAB    'NUMBER' = 'NUMBER' + 1
        JSR     $INCI

XBM090  RTS


;
; SOFT RESET (RESET KEY) COMMAND PROCESSOR -- CTRL-Z
;
XRESET  BEQ     XRS090

        JMP     RESTRT

XRS090  RTS
```

```
;
; COMMAND PROCESSOR FOR VARIABLE ITERATE -- #CVAR?<COMMAND>
;
XVAR    JSR     GETCH           GET VARIABLE NAME
        LDA     EXEC            EXECUTE MODE?
        BNE     XVA010          YES

        JMP     RCMD            NO -- SCAN PAST COMMAND & RETURN

XVA010  LDX     #NUMBER-DTAB    SAVE CURRENT VALUE OF 'NUMBER'
        JSR     SPSHI

        JSR     VFIND           FIND VARIABLE
        BEQ     XVA020          FOUND

        LDX     #NUMBER-DTAB    NOT FOUND --- USE ZERO
        JSR     SCLRI
        LDA     #ECUNDV         ... & FLAG SOFT ERROR
        STA     ERR
        BNE     XVA040          (BRA)

XVA020  LDY     #2              MOVE VARIABLE VALUE TO 'NUMBER'
        LDX     #0

XVA030  LDA     (FLINE),Y
        STA     NUMBER,X
        INX
        INY
        CPX     NUMBER-1        DONE?
        BNE     XVA030          NO

XVA040  JSR     GETCH           GET COMMAND TO ITERATE
        JMP     ITERB           GO TO COMMON CODE FOR ITERATION
```

```
;
; PLUS AND MINUS ARITHMETIC COMMAND PROCESSORS --- * X --
;
XPLUS   BEQ     XPL090          NON-EXECUTE
        LDX     #ACC-DTAB       INCREMENT 'ACC'
        JSR     SINCI
XPL090  RTS

XMINUS  BEQ     XMI090          NON-EXECUTE
        LDX     #ACC-DTAB       DECREMENT 'ACC'
        JSR     SDCRI
XMI090  RTS

XZERO   BEQ     XZE090          NON-EXECUTE
        LDX     #ACC-DTAB       ZERO 'ACC'
        JSR     SCLRI
XZE090  RTS
```

```
;
; TWHAT -- INFORMATION DUMP COMMAND PROCESSOR
;
TWHAT   DEQ     CURSOR          NON-EXECUTE
;
        LDA     MODE            DO NOTHING IF FULL GRAPHICS
        CMP     #MDRAW
        BEQ     XWH030
;
        CMP     #NORMAL         NORMAL MODE?
        BNE     XWH005          NO
;
        LDA     #EOL            YES -- ECHO EOL TO SCREEN (TEXT AREA)
        JSR     COUT
        JMP     XWH030              & DO NOTHING ELSE
;
XWH005  LDA     #CLEAR          CLEAR SCREEN BEFORE OUTPUTTING
        JSR     COUT
;
        LDY     #TWHAT-TWHAT    MESSAGE TABLE INDEX
        LDX     MODE
        LDA     TTOP,X
        CMP     #2              SPLIT SCREEN?
        BNE     TWH010          NO -- FULL DUMP IS FINE
;
        LDY     #TSWHAT-TWHAT   YES -- SMALL DUMP IS IN ORDER
;
XWH010  JSR     SCNWRT          OUTPUT DATA TO SCREEN
;
        LDA     MODE            PUT USER FUNCTIONS OUT IF APPROP
        CMP     #MDEBUG
        BNE     XWH030          NO
;
        LDA     #VLINES*2+7     SET CURSOR
        STA     ROWCRS
        LDA     #0
        STA     COLCRS+1
        LDA     LMARGN
        STA     COLCRS
        JSR     SETUDF          SET POINTER TO USER DEFINITIONS
;
XWH025  LDY     #0
;
XWH027  LDA     (FLINE),Y       GET DATA
        INY
        CMP     #$FF            END OF TABLE?
        BEQ     XWH030          YES -- DONE WITH USER DEFS
;
        PHA
        JSR     COUT            NO -- OUTPUT MORE DATA
        PLA
        CMP     #EOL            END OF DEFINITION?
        BNE     XWH027          NO -- KEEP PRINTING?
;
        LDY     #LINSIZ         YES -- BUMP TO START OF NEXT ONE
        LDX     #FLINE-DTAB
        JSR     FADDY
        JMP     XWH025
;
XWH030  LDY     #TPRMT-TWHAT    GET CURSOR BACK TO PROMPT
        JSR     SCNWRT
        LDA     PROMPT          SEE IF PROMPT IS BLANK
        CMP     #*                  WHICH MEANS EXECUTING
        BEQ     XWH050          YES
;
        LDA     #DELSPA         NO -- END-OF-EXECUTION
        JSR     COUT                FORCE CURSOR TO SHOW
;
XWH050  RTS
```

```
XWHOSO  LDY     #FRMT-THMAT     GET CURSOR BACK TO PROMPT
        JSR     SCNWRT
        LDA     PROMPT          SEE IF PROMPT IS BLANK
        CMP     #' '                WHICH MEANS EXECUTING
        BEQ     XWHOSO          YES

        LDA     #DELCHR         NO --- END OF EXECUTION
        JSR     DOUT                FORCE CURSOR TO SHOW

XWHOSO  RTS
```

```
;
;  MESSAGE TABLE FOR 'WHAT' COMMAND
;
;     FIVE BYTES PER ENTRY:
;
;        0 - ENTRY TYPE (#FF = END OF TABLE)
;        1 - CURSOR COLUMN (RELATIVE TO LEFT MARGIN)
;        2 - CURSOR ROW
;        3 - TEXT POINTER LSD
;        4 - TEXT POINTER MSD
;
TWHAT   BYTE   2,2,4          SCANNED INPUT LINE (DEBUG MODE STARTS HERE)
        WORD   LININ-1
        BYTE   0,0,6          USER VARIABLES
        WORD   VDEF-1
;
TSWHAT  BYTE   0,2,0          "ACC=" (SPLIT SCREEN START HERE)
        WORD   MACC
        BYTE   0,12,0         "NUMBER="
        WORD   MNUM
        BYTE   0,25,0         "LEVEL="
        WORD   MLEV
        BYTE   0,2,1          "CHAR="
        WORD   MCHR
;
        BYTE   0,25,1         "ERROR="
        WORD   MERR
        BYTE   0,21,1         ERROR CODE
        WORD   ERR-1
;
TVARS   BYTE   0,6,0          ACC VALUE
        WORD   ACC-1
        BYTE   0,19,0         NUMBER VALUE
        WORD   NUMBER-1
        BYTE   0,31,0         LEVEL VALUE
        WORD   LEVEL-1
        BYTE   0,7,1          CHAR VALUE
        WORD   CHAR-1
;
        BYTE   #FF            END OF TABLE
;
TPRMT   BYTE   0,0,2          SCREEN ENTRY AREA PROMPT
        WORD   PROMPT-1
;
        BYTE   #FF            END OF TABLE
;
        *=*+30                SPARES FOR PATCHING
;
;
MACC    BYTE   4,"ACC="
;
MNUM    BYTE   7,"NUMBER="
;
MLEV    BYTE   6,"LEVEL="
;
MCHR    BYTE   5,"CHAR="
;
MERR    BYTE   6,"ERROR="
;
;
;
```

```
;
; XSPEED -- SPEED CONTROL COMMAND PROCESSOR -- CTRL-B <NUMBER>
;
XSPEED  JSR     GETCH           GET CHARACTER THAT FOLLOWS
        LDA     EXEC            SEE IF SCAN OR EXECUTE
        BEQ     XSP090          NON-EXECUTE
;
        LDA     CHAR            EXECUTE -- CHANGE SPEED
        AND     #$07
        STA     SPEED
;
XSP090  RTS
;
;
; EDGE RULE SELECT COMMAND PROCESSOR -- CTRL-E <NUMBER>
;
XERULE  JSR     GETCH           GET CHARACTER THAT FOLLOWS
        LDA     EXEC            SEE IF SCAN OR EXECUTE
        BEQ     XER090          NON-EXECUTE
;
        LDA     CHAR            EXECUTE -- CHANGE EDGE RULE
        AND     #$03
        STA     EDGRUL
;
        JSR     CTEST           SEE IF TURTLE IN SCREEN LIMITS
        BEQ     XER090          YES -- NO PROBLEM
;
        JSR     XHOME+2         NO -- HOME AS A PRECAUTION
;
XER090  RTS
;
;
; MODE SELECT COMMAND PROCESSOR -- CTRL-M <NUMBER>
;
XMODE   JSR     GETCH           GET CHARACTER THAT FOLLOWS
        LDA     EXEC            SEE IF SCAN OR EXECUTE
        BEQ     XMD090          NON-EXECUTE
;
        LDA     CHAR            EXECUTE -- CHANGE MODE
        AND     #$03
        STA     MODE
        JSR     XHOME+2         ENSURE THAT CURSOR IS IN SCREEN LIMITS
        LDA     NXTSCN          BRING FORWARD NEXT MODE TO CURRENT
        STA     SCNMOD
        JSR     MODSEL
;
XMD090  RTS
;
;
; SCREEN MODE SELECT COMMAND PROCESSOR -- CTRL-O <NUMBER>
;
XDSPMD  JSR     GETCH           GET CHARACTER THAT FOLLOWS
        LDA     EXEC            SEE IF SCAN OR EXECUTE
        BEQ     XDM090          NON-EXECUTE
;
        LDA     CHAR            EXECUTE -- SET NEW MODE
        AND     #$07
        STA     NXTSCN          0-7 MAPS TO 1-8 LATER
```

```
XDR090  RTS

;
; XTREP -- TURTLE REPRESENTATION COMMAND PROCESSOR -- CTRL-T <NUMBER>
;
XTREP   JSR     GETCH           GET CHARACTER THAT FOLLOWS.
        LDA     EXEC            SEE IF SCAN OR EXECUTE.
        BEQ     XTR090          NON-EXECUTE.

        LDA     CHAR            EXECUTE -- CHANGE TURTLE REPRESENTATION.
        AND     #$03
        STA     TRTREP

        JSR     TRONOF          SETUP HARDWARE FOR NEW OPTION.

XTR090  RTS

;
; XAUDIO -- AUDIO SELECT COMMAND PROCESSOR -- CTRL-W <NUMBER>
;
XAUDIO  JSR     GETCH           GET CHARACTER THAT FOLLOWS.
        LDA     EXEC            SEE IF SCAN OR EXECUTE.
        BEQ     XAU090          NON-EXECUTE.

        LDA     CHAR            EXECUTE.
        AND     #$0F
        STA     AUDIO
        STA     AUDF1

XAU090  RTS
```

```
; ORIENTATION COMMANDS FOR TURTLE

;
; FACE NORTH COMMAND PROCESSOR -- H
;
XNORTH  BEQ     XRO090          NON-EXECUTE
        LDA     #0              SET INDEX TO NORTH (ZERO)
        BEQ     XRO020          (BRA)
;
; ROTATE RIGHT COMMAND PROCESSOR -- R
;
XROTR   BEQ     XRO090          NON-EXECUTE
        INC     ORIENT          EXECUTE -- BUMP INDEX
        BNE     XRO010          (BRA)
;
; ROTATE LEFT COMMAND PROCESSOR -- L
;
XROTL   BEQ     XRO090          NON-EXECUTE
        DEC     ORIENT          EXECUTE -- DECREMENT INDEX
XRO010  LDA     ORIENT          MASK RESULT TO 3 BITS
        AND     #$07
XRO020  STA     ORIENT
        JSR     PLCTRT          RE-ORIENT TURTLE REPRESENTATION
XRO090  RTS
;
; HOME COMMAND PROCESSOR -- H
;
XHOME   BEQ     XHM010          NON-EXECUTE
        LDA     #0              HOME = 0,0
        STA     XCURS
        STA     XCURS+1
        STA     YCURS
        STA     YCURS+1
        JSR     PLCTRT          PLACE TURTLE REPRESENTATION
        JSR     PLTPNT          LEAVE TRACK ALSO
XHM010  RTS
;
; CLEAR SCREEN COMMAND PROCESSOR -- C
;
XCLEAR  BEQ     XCL090          NON-EXECUTE
        LDA     IOCB2+ICHID     "S:" OPEN?
        BMI     XCL090          NO
        LDA     #CLEAR          YES -- SEND CLEAR SCREEN
        JSR     TOUT
```

```
            JSR     TSTPLT          PLOT POINT IF IN LIMITS
XCL090      RTS
```

```
; TURTLE SENSING COMMANDS

; EDGE SENSING COMMAND PROCESSOR -- E<THEN><ELSE>

XEDGE   BNE     XEDG10          EXECUTE

        JMP     SKIP            NON-EXECUTE

XEDG10                          TEST FOR MODE WHERE ONE COLOR IS TO BE
;                               TREATED THE SAME AS THE EDGE

        JSR     FTEST           TEST PIXEL IN FRONT OF TURTLE
        AND     #$01            RESULT OF OPERATION <> 0 IF BEYOND EDGE
        JMP     TEST



; COLOR SENSING COMMAND PROCESSOR -- S  (COLOR GOES TO 'ACC')

XSENSE  BEQ     XSND90          NON-EXECUTE

        LDA     IOCB2+ICHID     SEE IF IN A GRAPHICS MODE
        BMI     XSND05          NO -- IOCB2 IS CLOBBED

        JSR     FTEST           SEE IF POINT IN FRONT OF TURTLE IS IN SCREEN LIMITS
        BEQ     XSND10          YES -- COLOR IS SENSE-ABLE

XSND05  LDA     #0              NO -- RETURN "BACKGROUND" VALUE
        BEQ     XSND20          (BRA)

XSND10  LDX     #XCURS-DTAB     SAVE X & Y CURSOR VALUES
        JSR     SPSHT

        JSR     CFRWRD          MOVE CURSOR FORWARD
        JSR     SETCUR

        LDX     #ACC-DTAB       CLEAR 'ACC'
        JSR     SCLR1
        JSR     TIN             GET VALUE OF COLOR UNDER TURTLE
        AND     #$07
        STA     XSTEMP          SAVE TEMPORARILY.

        LDX     #XCURS-DTAB     RESTORE CURSOR X & Y VALUES
        JSR     SPULI
        LDA     XSTEMP          RESTORE COLOR SENSE VALUE.

; *** EXTERNAL ENTRY POINT ***

XSND20  CLC                     CONVERT TO ASCII
        ADC     #'0
        LDX     ACC-1           & STORE IN LSD OF 'ACC'
        STA     ACC-1,X

XSND90  RTS



; PEN CONTROL COMMAND PROCESSOR -- P  ('ACC' GOES TO 'COLOR')
```

```
XPEN    BEQ     XPNO90          NON-EXECUTE
        LDX     #ACC+0,X        CONVERT ACC TO BINARY
        JSR     GETBYT
        AND     #$7F
        BNE     ILLQTY          ILLEGAL QUANTITY ERROR
        BIT     XTRACT          NO -- O.K.
        LDA     #0,X            YES --- SKIP() ARGUMENT
XPNO10  BTS     COLORN          RESET UP/DOWN FLAG
        AND     #$7F
        EOR     COLORN
        STA     COLORN
        BMI     XPNO90          PEN-UP.
        JSR     TSTPLT          PLOT POINT IF TURTLE IN LIMITS
XPNO90  RTS


; PEN UP COMMAND PROCESSOR -- U

XUP     BEQ     XUPO90          NON-EXECUTE
        LDA     #$80            SIGN BIT IS FLAG FOR PEN UP
        ORA     COLORN
        STA     COLORN
XUPO90  RTS


; PEN DOWN COMMAND PROCESSOR -- D

XDOWN   BEQ     XDNO90          NON-EXECUTE
        LDA     #$7F            SIGN BIT IS FLAG FOR UP/DOWN
        AND     COLORN
        STA     COLORN
        JSR     TSTPLT          PLOT POINT IF TURTLE IN LIMITS
XDNO90  RTS


; TURTLE-BEEP COMMAND PROCESSOR -- B

XBEEP   BEQ     XBPO90          NON-EXECUTE
        LDA     #$50            FREQUENCY = ????HZ
        STA     AUDF1
        LDA     #$A0+8          TYPE = ???/VOLUME = 1+8
        STA     AUDC1
        LDY     #128            DELAY OUTER LOOP CONTROL
        LDX     #0              INNER LOOP = 256
```

```
            STA     AUDC1

            LDA     #$20            DELAY OUTER LOOP CONTROL
            LDY     #0              INNER LOOP = 256

XDFD1C      DEY
            BNE     XDFD1C

            DEX
            BNE     XDFD1C

            STX     AUDC1           TURN OFF TONE (PO)

XDFD9C      RTS


; JOYSTICK TEST COMMAND PROCESSOR -- #CLETTER> THEN-<ELSE>

XJOYS       JSR     GETCH           GET CHARACTER THAT FOLLOWS
            LDA     EXEC
            BNE     XJOY1C          EXECUTE

            JMP     SKIP            SCAN MODE -- SCAN BOTH THEN & ELSE COMMANDS

XJOY1C      LDA     CHAR            GET JOYSTICK SELECTION
            CMP     #'@             '@ TO 'A ARE TRIGGERS
            BCS     XJOY3C          TRIGGER TEST

            SEC                     JOYSTICK TEST -- NORMALIZE SELECT CHARACTER
            SBC     #A
            AND     #$03            USE LOWER BITS TO SELECT BIT MASK
            TAX

            LDA     CHAR            USE UPPER BITS TO SELECT JOYSTICK #
            SEC                     (CLEAR BORROW)
            SBC     #1
            LSR     A
            LSR     A
            AND     #$03
            TAY

            LDA     STICK0,Y        GET JOYSTICK DATA
            EOR     #$FF            DATA IS INVERTED -- CORRECT IT
            AND     BMASK,X         MASK DOWN TO SINGLE BIT

            JMP     TEST

XJOY3C      SBC     #'@             NORMALIZE SELECT CHARACTER
            AND     #$0F
            TAX
            LDA     PTRIG0,X
            EOR     #$FF            DATA IS INVERTED -- CORRECT IT
            AND     #$01
            JMP     TEST

BMASK       BYTE    $01, $08, $02, $04      F,R,B,L


; READ POT CONTROLLER TO ACCUMULATOR -- X<NUMBER>

XPOT        JSR     GETCH           GET CHARACTER THAT FOLLOWS
            LDA     EXEC
            BEQ     XPOT9C          NON-EXECUTE

            LDX     CHAR            GET POT SELECTION
```

```
            AND     #$07
            TAY

            LDA     #$0F            RESULT = $08 - PADDLE READING
            SEC                     (CLEAR BORROW)
            SBC     PADDLE,Y

            LDX     #ACC-DTAB       CONVERT RESULT TO ASCII NUMBER
            JSR     BITAJ               & STORE IN 'ACC'

    XPTOPD  RTS


;
; COLOR REGISTER UPDATE COMMAND PROCESSOR -- &(NUMBER);
;
    XCOLOR  JSR     RETCH           GET CHARACTER THAT FOLLOWS
            LDA     EXEC
            BEQ     XCOD90          NON-EXECUTE

            LDX     #ACC-DTAB       CONVERT 'ACC' TO BINARY
            JSR     BAIBI
            PHA                     SAVE RESULT

            LDA     CHAR            CALCULATE INDEX TO COLOR REGISTER
            AND     #$07
            CMP     #4+1            ONLY 0-4 VALID
            BCC     XCOD10          O.K.

            LDA     #'X             N.G.
            JMP     DIRECT

    XCOD10  TAY
            PLA                     GET COLOR/LUM VALUE
            STA     COLOR0,X            & STORE IN DATABASE

    XCOD90  RTS


;
; WAIT FOR NEXT CLOCK TICK COMMAND PROCESSOR -- W
;
    XWAIT   BEQ     XWAD90          NON-EXECUTE

            LDA     #1
            JSR     SYNSYN          SYNC TO CLOCK

    XWAD90  RTS


; SENSE TURTLE ORIENTATION COMMAND -- ;
;
    XCMPAS  BEQ     XCPD90          NON-EXECUTE

            LDX     #ACC-DTAB       SET 'ACC' TO ZERO
            JSR     SCLRI

            LDA     ORIENT          THEN SET LSD TO ORIENTATION #
            JMP     XSNO20

    XCMD90  RTS


; SET & SENSE GRAPHICS MODE COMMAND PROCESSOR -- G
```

```
XFR099  RTS

        TURTLE FORWARD COMMAND PROCESSOR -- F

; HANDLES EDGE RULES FOR STOP AT EDGE, REBOUND AT EDGE, WRAP
; AT EDGE AND DISAPPEAR AT EDGE

        ALSO HANDLES PEN UP OR DOWN

IFWARD  BEQ     XFRQ17          NON-EXECUTE

XFR010  JSR     ITEST           TEST FOR EDGE IN FRONT OF TURTLE
        BNE     XFR020          PAST EDGE

        JSR     CFWARD          MOVE CURSOR (TURTLE) FORWARD

XFR010  JSR     PLOTRT          PLACE TURTLE REPRESENTATION

XFR015  JSR     PLTFWT          LEAVE TURTLE TRACK (IF VALID)

XFR017  RTS


XFR020  LDA     EDGRUL          OFF EDGE -- WHAT IS CURRENT EDGE RULE?
        CMP     #ERSTOP         STOP?
        BEQ     XFR017          YES -- LEAVE TRACK WITHOUT MOVING

        CMP     #ERDISA         DISAPPEAR OFF EDGE?
        BEQ     XFR030          YES -- DO OFF EDGE

        CMP     #ERWRAP         WRAP SCREEN?
        BEQ     XFR040          YES -- DO CALCULATION

    REFLECT OFF WALL

        LDA     #EDGE           NO -- MUST BE REFLECT (BY DEFAULT)
        ORA     #EDGE           E/W WALL HIT?
        BEQ     XFR025          NO -- CHECK FOR N/S

        LDA     #8              YES -- EAST OR WEST WALL COLLISION
        SEC                     (CLEAR BORROW)
        SBC     ORIENT          'ORIENT' = 8 - 'ORIENT'
        STA     ORIENT

XFR025  LDA     #EDGE           N/S WALL HIT?
        ORA     #EDGE
        BEQ     XFR029          NO

        LDA     #12             YES -- NORTH OR SOUTH WALL COLLISION
        SEC                     (CLEAR BORROW)
        SBC     ORIENT          'ORIENT' = (12 - 'ORIENT') MOD 8
        AND     #$07
        STA     ORIENT

XFR029  JMP     XFR010          FINISH PROCESSING
  NOTE  ABOVE CODE WILL LOOP INDEFINITELY IF CURSOR GETS OUTSIDE
        OF EDGE OF SCREEN

    DISAPPEAR (WANDER)

XFR030  JSR     CFWARD          MOVE TURTLE BUT LEAVE NO TRACKS
        JSR     PLOTRT          REMOVE REP AS TURTLE GOES OFF SCREEN
        RTS

    WRAP SCREEN

XFR040  LDA     #EDGE           N/S HIT? WHAT
```

```
         ORA   SEDGE
         BEQ   XFR042          NO

         LDA   #-1             YES -- COMPLEMENT Y CURSOR
         SEC                   (CLEAR BORROW)
         SBC   YCURS
         STA   YCURS
         LDA   #-1             (NON-SYMMETRICAL SCREEN)
         SBC   YCURS+1
         STA   YCURS+1
         JMP   XFR045

XFR042   LDY   ORIENT          NO WRAP -- INCREMENT NORMALLY
         LDX   #YCURS-DTAB
         LDA   DYTAB,Y
         JSR   FDRWRD

XFR045   LDA   WEDGE           E/W WALL WRAP?
         ORA   EEDGE
         BEQ   XFR047          NO

         LDA   #-1             YES -- COMPLEMENT X CURSOR
         SEC                   (CLEAR BORROW)
         SBC   XCURS
         STA   XCURS
         LDA   #-1             (NON-SYMMETRICAL SCREEN)
         SBC   XCURS+1
         STA   XCURS+1
         JMP   XFR050

XFR047   LDY   ORIENT          NO WRAP -- INCREMENT NORMALLY
         LDX   #XCURS-DTAB
         LDA   DXTAB,Y
         JSR   FDRWRD

XFR050   JMP   XFR012          PLACE TURTLE.
```

```
;
; GET USER DEFINITIONS FROM DEVICE -- CTRL-A ' <DEVICE SPEC>'
;
XGETFL  JSR     DNAME           SCAN TO END OF DEVICE SPECIFICATION
        LDA     EXEC
        BEQ     XGFO90          NON-EXECUTE

        LDA     #DREAD          OPEN DEVICE FOR INPUT
        JSR     DOPEN

        JSR     CLRUDF          CLEAR CURRENT USER DEFINITION AREA

        JSR     SETUDF          SETUP POINTER TO USER DEFS

XGFO10  LDY     #0

XGFO1S  JSR     DINP            READ A DATA BYTE
        CMP     #EOF            END OF FILE?
        BEQ     XGFO30          YES

        STA     (FLINE),Y       NO -- STORE DATA
        INY
        CMP     #EOL            END OF A DEFINITION?
        BNE     XGFO1S          NO

        LDY     #LINSIZ         YES -- BUMP ADDRESS TO START OF NEXT
        LDX     #FLINE-DTAB
        JSR     FADDY
        JMP     XGFO1S

XGFO30  JSR     DCLOSE          CLOSE OPEN DEVICE

XGFO90  RTS
```

```
;
; PUT USER DEFINITIONS TO DEVICE == CTRL-P " <DEVICE SPEC> "
;
XPUTFL  JSR     DNAME           SCAN TO END OF DEVICE SPECIFICATION.
        LDA     EXEC
        BEQ     XPF090          NON-EXECUTE.

        LDA     #OWRIT          OPEN DEVICE FOR OUTPUT.
        JSR     DOPEN

        JSR     SETUDF          SETUP POINTER TO USER DEFINITION AREA.

XPF010  LDY     #0
        LDA     (FLINE),Y       GET A DATA BYTE.

XPF015  CMP     #$FF            END OF TABLE?
        BEQ     XPF030          YES -- DONE.

        CMP     #EOL            END OF A DEFINITION?
        BEQ     XPF020          YES.

        LDA     (FLINE),Y       GET DEFINITION DATA.
        INY
        PHA
        JSR     DOUT            OUTPUT TO DEVICE.
        PLA
        JMP     XPF015

XPF020  LDY     #LINBIZ         BUMP POINTER TO NEXT DEFINITION START.
        LDX     #FLINE-DTAB
        JSR     PADDY
        JMP     XPF010

XPF030  LDA     #EOF            PUT END OF FILE AT END.
        JSR     DOUT
        JSR     DCLOSE          CLOSE THE OPEN FILE.

XPF090  RTS
```

```
; LOAD PREDEFINED COMMANDS FROM ROM COMMAND -- CTRL-L <CHAR>

XLOAD   JSR     GETCH           GET CHARACTER THAT FOLLOWS
        LDA     EXEC
        BEQ     XLOD90          NON-EXECUTE

; *** EXTERNAL ENTRY POINT ***

XLOAD2  JSR     CLRUDF          CLEAR CURRENT USER DEFINITION AREA

        JSR     SETUDF          SET POINTERS TO USER AREA.

        LDX     #0              INITIALIZE NAME TABLE INDEX

XLOD10  LDA     LODTAB,X        SCAN TABLE FOR MATCH
        CMP     #$FF            END OF TABLE?
        BNE     XLOD30          NO

        LDA     #ECLOAD         YES -- ARGUMENT ERROR
        JMP     DIRECT

XLOD30  CMP     CHAR            MATCH FOUND?
        BEQ     XLOD50          YES

        INX                     NO -- GO TO NEXT ENTRY.
        INX
        INX
        INX
        JMP     XLOD10

XLOD50  LDA     LODTAB+2,X      SETUP POINTER TO CANNED DEFINITIONS
        STA     TEMP
        LDA     LODTAB+3,X
        STA     TEMP+1
        LDA     (LODTAB+1),X    GET TOP LEVEL COMMAND NAME
        STA     TEMP+2

        LDY     #0              GET SET TO MOVE DEFINITIONS.

XLOD55  LDA     #0              START OF NEW DEFINITION.
        STA     FLINE+2

XLOD60  LDA     (TEMP),Y        GET A CHARACTER.
        CMP     #EOF            END OF FILE?
        BEQ     XLOD90          YES -- DONE.

        INY                     NO -- MOVE DATA.
        STY     COUNT           SAVE SOURCE INDEX
        LDY     FLINE+2         DESTINATION INDEX.
        STA     (FLINE),Y
        INY
        STY     FLINE+2         SAVE DESTINATION INDEX
        LDY     COUNT           SOURCE INDEX.
        CMP     #EOL            END OF A SINGLE DEFINITION?
        BNE     XLOD60          NO.

        LDY     #LINSIZ         YES -- DUMP ADDRESS TO START OF NEXT
        LDA     #FLINE-DTAB
        JSR     PADDY
        LDY     COUNT           SOURCE INDEX
        JMP     XLOD55

XLOD90  RTS
```

```
;
; LOAD AND RUN CANNED DEFINITIONS FROM ROM -- CTRL-R <CHARACTER>.
;
XRUN    JSR     GETCH           GET CHARACTER THAT FOLLOWS.
        LDA     EXEC
        BEQ     XRN090          NON-EXECUTE.

; *** EXTERNAL ENTRY POINT ***

XRUN2   JSR     XLOAD2          LOAD CANNED DATA.

        LDA     TEMP+2          GET TOP LEVEL COMMAND NAME.
        STA     CHAR
        JSR     CMD             EXECUTE IT.

XRN090  RTS
```

```
;
; START OF LEVEL 3 ROUTINES -- SPECIAL PURPOSE UTILITIES
;
;
; NUMB -- SCAN INPUT TO END OF NUMERIC FIELD
;
; CALLING SEQUENCE:
;
;       'CHAR' = 1ST NUMERIC DIGIT
;
;       JSR     NUMB
;
;       'NUMBER' = VALUE OF NUMERIC FIELD (IF INPUT TOO LONG, USES LAST N DIGITS)
;       'CHAR'  = CHARACTER AFTER END OF NUMERIC FIELD
;
NUMB    LDX     #NUMBER-DTAB    ZERO NUMBER FIRST
        JSR     SCLR1
        LDX     ACC-1           GET RECORD LENGTH
;
NUMB20  LDA     CHAR            STORE FIRST DIGIT IN LSD.
        STA     NUMBER-1,X
        JSR     GETCH           GET NEXT CHARACTER
        JSR     DECDIG          DECIMAL DIGIT?
        BCS     NUMB24          YES
;
        RTS                     NO -- ALL DONE.
;
NUMB24  LDY     #0              SHIFT DIGITS ONE TO LEFT.
;
NUMB30  LDA     NUMBER+1,X      SHIFT LEFT ONE DIGIT
        STA     NUMBER,X
        INX
        CPX     NUMBER-1        (INTENTIONALLY MOVES ONE TOO MANY).
        BNE     NUMB30
;
        BEQ     NUMB20          ADD NEXT DIGIT
```

```
;  CFRWRD -- MOVE CURSOR 'TURTLE' FORWARD
;
;  CALLING SEQUENCE:
;
;        'ORIENT' = ORIENTATION VALUE (0-3)
;        'XCURS'  = CURSOR X POSITION
;        'YCURS'  = CURSOR Y POSITION
;
;        JSR      CFRWRD
;
;        'XCURS' = 'XCURS' + 'DXTAB'('ORIENT')
;        'YCURS' = 'YCURS' + 'DYTAB'('ORIENT')
;
CFRWRD  LDY      ORIENT          GET ORIENTATION
        LDX      #XCURS-DTAB     X POSITION FIRST
        LDA      DXTAB,Y         GET INCREMENT
        JSR      FORWRD          ADJUST POSITION
;
        LDX      #YCURS-DTAB     THEN Y POSITION.
        LDA      DYTAB,Y         GET INCREMENT
;
;  *** EXTERNAL ENTRY POINT ***
;
FORWRD  BEQ      CFR090          NO CHANGE.
;
        BPL      CFR0&0          +1.
;
        JMP      DDCR1           -1 & RETURN
;
CFR0&0  JMP      DINC1           +1 & RETURN
;
CFR090  RTS
;
;  X & Y INCREMENT TABLES (INDEXED BY 'ORIENT')
;
DYTAB   .BYTE    $FF,$FF         (NOTE: THIS TABLE OVERLAPS 'DXTAB'!!!)
;
DXTAB   .BYTE    $00,$01,$01,$01,$00,$FF,$FF,$FF
```

```
;
; FTEST -- TEST FOR EDGE IN FRONT OF TURTLE
;
; CALLING SEQUENCE:
;
;       'XCURS' & 'YCURS' = CURSOR VALUES
;
;       JSR     FTEST
;       BEQ     IN BOUNDS (A = $00)
;       ELSE OUT OF BOUNDS (A = $01)
;
;       SEE ALSO 'EDGTST'
;
FTEST   LDY     #XCURS-DTAB     SAVE X & Y CURSOR
        JSR     SPSHJ
;
        JSR     CFWRD           MOVE TURTLE FORWARD
;
        JSR     CTEST           TEST FOR EDGE
        STA     FTSTAT          SAVE EDGE TEST STATUS
;
        LDY     #XCURS-DTAB     RESTORE X & Y CURSOR
        JSR     SPULJ
;
        LDA     FTSTAT          RESTORE STATUS & CC
        RTS                     RETURN WITH CC SET
;
;
; CTEST -- TEST FOR EDGE UNDER CURSOR (TURTLE)
;
; CALLING SEQUENCE:
;
;       (SEE 'FTEST' AND 'EDGTST')
;
CTEST   JSR     EDGTST          PERFORM EDGE BOUNDS TEST
;
        LDA     NEDGE           SEE IF ALL IN BOUNDS
        ORA     SEDGE
        ORA     #EDGE
        ORA     EEDGE
        RTS                     RETURN WITH A & CC SET
;
;
; EDGTST -- TEST FOR CURSOR ON EDGE OR OUT OF BOUNDS
;
; CALLING SEQUENCE:
;
;       'XCURS' & 'YCURS' = TURTLE LOCATION
;
;       JSR     EDGTST
;
;       'NEDGE', 'SEDGE', 'WEDGE' & 'EEDGE' SET TO REPRESENT
;       STATUS AT THE NORTH, SOUTH, WEST AND EAST WALLS
;           $00 = CURSOR IN BOUNDS
;           $01 = CURSOR OUT OF BOUNDS
;
EDGTST  LDY     #YCURS-DTAB     CHECK NORTH WALL
        LDX     #YMIN-XMIN
        JSR     CMPRNG
        LSR     A
        STA     NEDGE
```

```
        LDY     #YMAX-MMTAB     CHECK SOUTH WALL
        JSR     CHKRNG
        AND     #01
        STA     SEDGE

        LDX     #XCHRS-DTAB     CHECK WEST WALL
        LDY     #XMIN-MMTAB
        JSR     CHKRNG
        LSR     A
        STA     WEDGE

        LDY     #XMAX-MMTAB     CHECK EAST WALL
        JSR     CHKRNG
        AND     #01
        STA     EEDGE

        RTS


;
;  CHKRNG -- RANGE CHECK A SIGNED CURSOR COORDINATE WITH A MIN/MAX VALUE
;
;  CALLING SEQUENCE:
;
;       X = 'DTAB' INDEX TO CURSOR
;       Y = 'MMTAB' INDEX TO TABLE ENTRY (FURTHER INDEXED BY 'SCNMOD' INTERNALLY)
;
;       JSR     CHKRNG
;
;       A = $00 IF 'DTAB'(X) = 'MMTAB'(Y,'SCNMOD')
;           $01 IF 'DTAB'(X) > 'MMTAB'(Y,'SCNMOD')
;           $02 IF 'DTAB'(X) < 'MMTAB'(Y,'SCNMOD')
;           CC SET TO REFLECT REGISTER A VALUE
;
;       Y REGISTER IS CLOBBERED
;
CHKRNG  ASL     SCNMOD          X2 FOR INDEX
        TYA                     MODIFY TABLE INDEX
        ADC     SCNMOD
        TAY
        LSR     SCNMOD          RESTORE 'SCNMOD'
        LDA     DTAB+1,X        COMPARE CURSOR WITH TABLE ENTRY
        CMP     MMTAB+1,Y
        BEQ     CHR030          MSBS ARE EQUAL

        BPL     CHR050          CURSOR > CHECK VALUE

        BMI     CHR025          CURSOR < CHECK VALUE

CHR030  LDA     DTAB,X          CHECK LSBS
        SBC     MMTAB,Y
        BEQ     CHR040          CURSOR = CHECK VALUE

        BCS     CHR050          CURSOR > CHECK VALUE

CHR025  LDA     #$02            CURSOR < CHECK VALUE
        RTS

CHR040  LDA     #$00            CURSOR = CHECK VALUE
        RTS

CHR050  LDA     #$01            CURSOR > CHECK VALUE
        RTS                     RETURN WITH CC & A SET


; MIN/MAX TABLES FOR CURSOR (ORDERED BY HARDWARE SCREEN MODES 1-11)
; SEE ALSO 'SETCUR' & 'PLOTR1' FOR RELATED TABLES

MMTABS
```

```
; MIN/MAX TABLES FOR CURSOR (ORDERED BY HARDWARE SCREEN MODES 1-11)
; SEE ALSO 'SETCUR' & 'PLCTRT' FOR RELATED TABLES
;
MMTAB=*

XMIN    .WORD   -10,-10,-20,-40,-40,-80,-80,-160,-40,-40,-40

XMAX    .WORD   9,9,19,39,39,79,79,159,39,39,39

YMIN    .WORD   -12,-6,-12,-24,-24,-48,-48,-96,-96,-96,-96

YMAX    .WORD   11,5,11,23,23,47,47,95,95,95,95
```

```
;
; MODSEL -- OPERATING MODE I/O SELECTION
;
; CALLING SEQUENCE:
;
;       'MODE' = OPERATING MODE (0,1,2)
;       'SCRMOD' = SCREEN MODE SELECTION
;
;       JSR     MODSEL
;
;       SETS UP TEXT(0), G.I.(1) OR G.O.(2) FOR MODE
;
MODSEL  JSR     TRDOF           DISABLE TURTLE REF DURING CHANGES
        LDX     MODE            OPEN THE COMMAND INPUT DEVICE
        LDY     TINV,X
        JSR     SETSCN
;
        LDX     MODE            OPEN THE TEXT OUTPUT DEVICE
        LDY     TOTV,X
        JSR     SETSCN
;
        LDX     MODE            OPEN THE TURTLE GRAPHICS INPUT/OUTPUT DEVICE
        LDY     TIOV,Y
        JSR     SETSCN
;
        JSR     TRONOF          SETUP HARDWARE FOR TURTLE REF (ON OR OFF)
;
        JSR     TSTPLT          LEAVE A TURTLE TRACE IF TURTLE IN SCREEN LIMITS
;
        RTS
;
;
;
; SETSCN -- SETUP THE IOCB FOR ONE DEVICE
;           CLOSE THE IOCB;PUT IN NEW INFO, OPEN THE IOCB & SETUP FOR READ/WRITE
;
;  CALLING SEQUENCE:
;
;       Y = INDEX TO IOCB SETUP TABLES
;
;       JSR     SETSCN
;
;       X IS CLOBBERED
;
SETSCN  STY     TEMP            SAVE INDEX
        TYA                     REMOVE "CLOSE ONLY" INDICATOR
        AND     #$7F            (SIGN BIT)
        TAY
        LDX     TIO,Y           GET IOCB INDEX
        LDA     #CLOSE
        STA     ICCOM,X
        JSR     CIO             CLOSE THAT IOCB
;
;  RE-OPEN DEVICE IF SPECIFIED
;
        LDY     TEMP
        BMI     RTSORD          DEVICE NOT TO BE OPENED
;
        LDA     TDEV,Y          SETUP DEVICE NAME
        STA     OPNBUF
        LDA     #'              SET 'OPNBUF' TO "X:,EOL,0"
        STA     OPNBUF+1
        LDA     #EOL
        STA     OPNBUF+2
        LDA     #OPEN
        STA     ICCOM,X
        LDA     TAX1,Y          SETUP AUX1
```

```
          LDA     #'                SET 'OPNBUF' TO '# 'EOL'
          STA     OPNBUF+1
          LDA     #EOL
          STA     OPNBUF+2
          LDA     #OPEN
          STA     ICCOM,X
          LDA     TAX1,Y            SETUP AUX1
          STA     ICAUX1,X
          LDA     TAX2,Y            SETUP AUX2
          BEQ     ST5020            FORCE SCREEN MODE TO ZERO

          CLC                       SCREEN MODE = INTERNAL MODE + CONSTANT
          ADC     SCRMOD

ST5020    STA     ICAUX2,X

          LDA     SSTACK            LET SCREEN HANDLER KNOW CURRENT UPPER
          STA     APPMHI                BOUND
          LDA     SSTACK+1
          CLC
          ADC     #1                LEAVE ONE PAGE MARGIN
          STA     APPMHI+1

          JSR     CIO               OPEN THE IOCB

          LDY     TEMP
          LDA     TOP,Y             SETUP READ/WRITE OPERATION
          STA     ICCOM,X

ST5090    RTS


TDEV      BYTE    'E,'S,'E,'K,  E

TAX1      BYTE    OWRIT,OWRIT+OREAD+NOCLR,OWRIT+OREAD+SPLIT+NOCLR,OREAD,OREAD

TAX2      BYTE    0,1,1,0,0

TOP       BYTE    PUTC,PUTC,PUTC,GETC,GETC

TIO       BYTE    IOCB0,IOCB2,IOCB2,IOCB1,IOCB1

; SIGN BIT SET INDICATES IOCB TO BE CLOSED & NOT RE-OPENED

TINX      BYTE    3,4,3,3           COMMAND INPUT IOCB (IOCB 1 ) 'CHIN'

TOTX      BYTE    $80,0,0,0         COMMAND OUTPUT IOCB (IOCB 0) 'COUT'

TTDX      BYTE    1,$81,2,2         TURTLE I/O IOCB (IOCB 2) 'TIN' & 'TOUT'

; INDEX TO ABOVE  0 = DRAW MODE
;                 1 = DEBUG MODE
;                 2 = SPLIT SCREEN DEBUG MODE
;                 3 = NORMAL MODE (SPLIT SCREEN)
```

```
;
; SCNWRT --- WRITES DATA TO SCREEN FROM TABLE ENTRIES
;
; CALLING SEQUENCE:
;
;       'LMARGN' = LEFT MARGIN OFFSET
;       '(PDSPTB)' = POINTER TO DISPLAY TABLE
;       Y = DISPLAY TABLE INDEX
;
;       JSR     SCNWRT
;
;       Y IS CLOBBERED
;
; EACH DISPLAY TABLE ENTRY CONSISTS OF 3 BYTES AS FOLLOWS:
;
;       0 - ENTRY TYPE (=0 = END OF TABLE)
;       1 - CURSOR X POSITION (HARDWARE NOTATION)
;       2 - CURSOR Y POSITION (HARDWARE NOTATION)
;       3 - LSB OF ADDRESS OF DATA RECORD
;       4 - MSB OF ADDRESS OF DATA RECORD
;
SCNWRT  LDA     IOCB0+ICHID     SEE IF OUTPUT DEVICE IS OPEN.
        BMI     SCNOUT          DEVICE IS NOT OPEN.
;
        LDA     #$FF            DISABLE CURSOR DURING RANDOM OUTPUTTING
        STA     CRSINH
;
SCN005  LDA     (PDSPTB),Y      GET ENTRY TYPE
        CMP     #$FF            END OF TABLE?
        BNE     SCN010          NO
;
        INC     CRSINH          YES -- RE-ENABLE CURSOR (=0)
;
SCNOUT  RTS
;
SCN010  INY
        LDA     (PDSPTB),Y      SET CURSOR
        CLC
        ADC     LMARGN          (CORRECT FOR LEFT MARGIN)
        PHA                     (SAVE A)
        LDX     MODE            (DETERMINE WHICH CURSOR SET)
        LDA     TTDV,X
        CMP     #2              SPLIT SCREEN?
        BNE     SCN015
;
        PLA                     (RESTORE A)
        STA     SPTCOL              X POSITION
        LDA     #0
        STA     SPTCOL+1
        INY
        LDA     (PDSPTB),Y
        STA     SPTROW              & Y POSITION
        JMP     SCN017
;
SCN015  PLA                     (RESTORE A)
        STA     COLCRS              X POSITION
        LDA     #0
        STA     COLCRS+1
        INY
        LDA     (PDSPTB),Y
        STA     ROWCRS              & Y POSITION
;
SCN017  INY
        LDA     (PDSPTB),Y      MOVE DATA RECORD ADDRESS TO 'SNTEMP'
        STA     SNTEMP
        INY
        LDA     (PDSPTB),Y
        STA     SNTEMP+1
```

```
SCN017  INY
        LDA     (PDSPTB),Y      MOVE DATA RECORD ADDRESS TO 'SWTEMP'
        STA     SWTEMP
        INY
        LDA     (PDSPTB),Y
        STA     SWTEMP+1
        INY
        STY     SWTEMP+2        SAVE TABLE INDEX
        LDY     #0              PREPARE TO GET DATA FROM RECORD
        LDA     (SWTEMP),Y      GET RECORD LENGTH
        TAX

SCN020  INY                     BUMP TO NEXT BYTE
        LDA     (SWTEMP),Y      GET DATA
        JSR     COUT            OUTPUT TO DEVICE
        DEX                     DONE?
        BNE     SCN020          NO -- KEEP GOING

        LDY     SWTEMP+2        YES -- RESTORE DISPLAY TABLE INDEX
        JMP     SCN005          ... & PROCESS NEXT ENTRY
```

```
;
; SETCUR -- SET HARDWARE CURSOR
;
; CALLING SEQUENCE
;
;       'XCURS' & 'YCURS' = TURTLE CURSOR
;       'SCNMOD' = SCREEN MODE
;
;       JSR     SETCUR
;
;       'COLCRS' & 'ROWCRS' = HARDWARE CURSOR VALUES

SETCUR  LDA     IOCB2*ICHID     SEE IF OUTPUT DEVICE IS OPEN
        BMI     STCOPO          NOT OPEN -- DO NOTHING

        LDX     SCNMOD          GET SCREEN MODE (DETERMINES SIZE)
        CLC
        LDA     XCURS
        ADC     XCENTR,X        ADJUST FOR DIFFERENT ORIGINS
        STA     COLCRS
        LDA     XCURS+1
        ADC     #0
        STA     COLCRS+1

        CLC
        LDA     YCURS
        ADC     YCENTR,X        ADJUST FOR DIFFERENT ORIGINS
        STA     ROWCRS

STCOPO  RTS

; SCREEN CENTER TABLES FOR CURSOR (ORDERED BY SCREEN MODES 1-11)
; SEE ALSO 'CHKRNG' & 'PLCTRT' FOR RELATED TABLES

XCENTR  .BYTE   10,10,20,40,40,80,80,160,40,40,40

YCENTR  .BYTE   12,6,12,24,24,48,48,96,96,96,96
```

```
;
; TURTLE REPRESENTATION ROUTINES
;


; TRONOF -- TURN MISSILE DMA ON OR OFF
; CALLING SEQUENCE
;       'TRTREP' = 0 IF OFF; ELSE ON

TRONOF  LDA     IOCB0+ICHID     ; IS  OPEN?
        BMT     TROTOO          ; NO -- TURTLE REPRESENTATION OFF

        LDA     TRTREP          ; TURTLE REPRESENTATION SELECTED?
        BEQ     TROTOO          ; NO -- OFF

        TAX                     ; SET COLOR REGISTERS
        LDA     TCOLOR-1,X      ; GET COLOR FROM TABLE
        STA     PCOLR0
        STA     PCOLR1
        STA     PCOLR2
        STA     PCOLR3

        LDA     #GRANON         ; EVERYTHING ON ?  -- TURN HIM ON.
        STA     GRACTL

        LDA     DMACT
        ORA     #DMACON         ; ENABLE MISSILE DMA (LOW RESOLUTION MODE)
        STA     DMACT

        JSR     PLOTRT          ; PLACE TURTLE REPRESENTATION ON SCREEN

        RTS

TROFF== =                       ; *** EXTERNAL ENTRY POINT ***

TROTOO  LDA     DMACT           ; PLAYER DMA OFF
        AND     #$FF-DMACON
        STA     DMACT

        LDA     #0
        STA     GRACTL
        STA     GRAFM

        RTS


; PLOTRT -- PLACE TURTLE REPRESENTATION ON SCREEN
;
; CALLING SEQUENCE
;
;       'TRTREP'  = 0 IF DESELECTED; ELSE SELECTED
;       'MONMOD'  = CURRENT SCREEN MODE SELECTED
;       'ORIENT'  = CURRENT TURTLE ORIENTATION
;       'XCURS'   = TURTLE POSITION: X COORDINATE
;       'YCURS'   = TURTLE POSITION: Y COORDINATE
;
        JSR     PLOTRT
```

```
PLCTST  LDA     [DISPLIST],Y    GET ITEM
        BMI     PLCSKP          SKIP IF NO TURTLE
        CLC                     TURTLE REPRESENTATION SELECTED?
        BNE     PLCRTS          YES
PLCSKP  RTS
PLCOPN  LDX     #YPOS           GET OLD POSITION
        LDY     #0
        LDA     #0
PLCOP2  STA     TABLE,X         REMOVE OLD REPRESENTATION
        INX
        DEY
        BNE     PLCOP2
        JSR     CTEST           TURTLE ON SCREEN?
        BNE     PLCOPO          NO
    CONVERT CURSOR X TO COLOR CLOCKS
        JSR     SETCUR          CONVERT TURTLE CURSOR TO HANDLER COORDINATE SYSTEM
        LDY     SCRMOD          DEPENDS UPON SCREEN MODE
        LDY     COPATS-1        GET # OF COLOR CLOCKS PER X POSITION
        BEQ     PLCO30          1/2 CLOCK IS SPECIFIED BY 0 IN TABLE
        TYA
        CLC
        ROR     A               START WITH 1/2 POSITION OFFSET
        CLC
PLCO20  ADC     COLCRS          NOW GO MULTIPLY
        DEY
        BNE     PLCO20
        BEQ     PLCO40          (BRA)
PLCO30  LDA     COLCRS+1        DIVIDE BY 2 (1/2 COLOR CLOCK)
        ROR     A
        LDA     COLCRS
        ROR     A
PLCO40  CLC
        ADC     #$30            LEFT EDGE OFFSET
        LDY     ORIENT          SUBTRACT ORIENTATION OFFSET
        SEC                     (CLEAR BORROW)
        SBC     TROY,Y
        CLC
        STA     HPOSM0          RESULT IS MISSILE HORIZONTAL POSITION
        ADC     #2
        STA     HPOSM1
        ADC     #2
        STA     HPOSM2
    CONVERT CURSOR Y POSITION TO SCAN LINES
        LDY     SLPTS,X         GET # OF SCAN LINES PER Y POSITION
        BEQ     PLCOS3          1/2 CLOCK
        TYA
        CLC
        ROR     A               START WITH 1/2 POSITION OFFSET
        CLC
PLCOS0  ADC     ROWCRS          MULTIPLY
```

```
            DEY     PLC063              FIX COUNT

            TYA
            CLC
            ROR     A                   START WITH 1/2 POSITION OFFSET
            CLC

PLC060      ADC     ROWCRS              MULTIPLY
            DEY
            BNE     PLC060

            BEQ     PLC065              (BRA)

PLC063      LDA     ROWCRS              DIVIDE BY 2
            CLC
            ROR     A

PLC065      ADC     #TVBUFF-TRBUFF+4        *** MAGIC OFFSET ***
            LDY     ORIENT              SUBTRACT ORIENTATION OFFSET
            SEC                         (CLEAR BORROW)
            SBC     TRDY,Y
            STA     TRYPOS              SAVE FOR NEXT TIME IN
            TAY                         SETUP FOR NOW

            LDA     TRTREF              GET PATTERN FOR CURRENT SELECTION
            SEC                         (CLEAR BORROW)
            SBC     #1
            ASL     A                   X8
            ASL     A
            ASL     A
            ADC     ORIENT              GET PATTERN FOR CURRENT ORIENTATION
            ASL     A                   X8
            ASL     A
            ASL     A
            TAX                         INDEX TO TABLE OF PATTERNS

            LDA     #8                  # OF BYTES IN PATTERN
            STA     TEMP

PLC080      LDA     TURTLE,X            MOVE PATTERN
            STA     TRBUFF,Y                TO PLAYER BUFFER
            INX
            INY
            DEC     TEMP
            BNE     PLC080

PLC090      RTS
; TURTLE MISSILE CHARACTERISTICS (BY MODE)
; SEE 'CHARMG' & 'PLCTBT' FOR RELATED TABLES.

; SCAN LINES PER CURSOR VERTICAL UNIT (BY MODE 1-11)

SLFYTB  .BYTE   4,8,4,2,2,1,1,0,0,0,0   (0 = 1/2)

; COLOR CLOCKS PER HORIZONTAL UNIT (BY MODE 1-11)

CCPXTB  .BYTE   8,8,4,2,2,1,1,0,2,2,2   (0 = 1/2)

; ORIENTATION OFFSET VERTICAL (NOTE: TABLE OVERLAPS ONE THAT FOLLOWS)

TRDY    .BYTE   0,0

; ORIENTATION OFFSET HORIZONTAL DIRECTION

TRDX    .BYTE   5,6,6,6,3,0,0,0

; TITLE 'PLAYER-PLA ETC TYPE 5 INITIATION'
```

```
;
TURTLE = *
;
; ARROW TURTLE
;
TURTL1   .BYTE   $10,$38,$10,$10,$10,$10,$10,$00 N
         .BYTE   $08,$06,$08,$10,$20,$40,$80,$00 NE
         .BYTE   $00,$00,$04,$FE,$04,$00,$00,$00 E
         .BYTE   $80,$40,$20,$10,$08,$06,$06,$00 SE
         .BYTE   $10,$10,$10,$10,$10,$39,$10,$00 S
         .BYTE   $02,$04,$08,$10,$20,$C0,$C0,$00 SW
         .BYTE   $00,$00,$40,$FE,$40,$00,$00,$00 W
         .BYTE   $C0,$C0,$20,$10,$08,$04,$02,$00 NW
;
; TURTLE TURTLE
;
TURTL2   .BYTE   $10,$7C,$FE,$7C,$7C,$FE,$00,$00 N
         .BYTE   $39,$1E,$8E,$7F,$3F,$1D,$08,$04 NE
         .BYTE   $48,$7C,$7C,$7E,$7C,$7C,$48,$00 E
         .BYTE   $04,$08,$1D,$3F,$7F,$8E,$1E,$39 SE
         .BYTE   $00,$FE,$7C,$7C,$FE,$7C,$10,$00 S
         .BYTE   $20,$10,$B8,$FC,$FE,$7D,$7B,$9C SW
         .BYTE   $24,$7C,$7C,$FC,$7C,$7C,$24,$00 W
         .BYTE   $9C,$7B,$7D,$FE,$FC,$B8,$10,$20 NW
;
; POINT TURTLE
;
TURTL3   .BYTE   $10,$00,$00,$00,$00,$00,$00,$00 N
         .BYTE   $02,$00,$00,$00,$00,$00,$00,$00 NE
         .BYTE   $00,$00,$00,$02,$00,$00,$00,$00 E
         .BYTE   $00,$00,$00,$00,$00,$00,$02,$00 SE
         .BYTE   $00,$00,$00,$00,$00,$00,$10,$00 S
         .BYTE   $00,$00,$00,$00,$00,$00,$80,$00 SW
         .BYTE   $00,$00,$00,$80,$00,$00,$00,$00 W
         .BYTE   $80,$00,$00,$00,$00,$00,$00,$00 NW
;
; TURTLE COLOR/LUM FOR EACH REPRESENTATION
;
TCOLOR   .BYTE   $0E,$E4,$0E
```

```
;
; TSTPLT -- PLOT POINT IF TURTLE IN SCREEN LIMITS
;
;  CALLING SEQUENCE:
;
;       JSR     TSTPLT
;
TSTPLT  JSR     CTEST           SEE IF TURTLE IN LIMITS
        BNE     PLTO90          NO -- DON'T PLOT
;                               YES -- FALL THROUGH TO 'PLTPNT'
;
;
;  PLTPNT -- PLOT POINT (LEAVE TURTLE TRACK) IF VALID
;
;  CALLING SEQUENCE:
;
;       (COLORN  = CURRENT PEN COLOR (#80 = PEN UP)
;
;       JSR     PLTPNT
;
;       NOTE:  ASSUMES THAT THE CURSOR IS IN SCREEN LIMITS!!!
;
PLTPNT  LDA     COLORN          SEE IF PEN DOWN
        BMI     PLTO90          NO -- UP
;
        LDA     IOCB2+ICHID     SEE IF IN A GRAPHICS MODE
        BMI     PLTO90          NO
;
        JSR     SETCUR          O.K.  -- ESTABLISH CURSOR
        LDA     COLORN          NOW PLOT POINT
        JSR     TOUT
;
PLTO90  RTS
;
;
;
;  CLKSYN -- CLOCK SYNCHRONIZATION ROUTINE
;
;  CALLING SEQUENCE:
;
;       A = DELAY FACTOR
;
;       JSR     CLKSYN
;
;       RETURNS ONLY AFTER CLOCK VALUE CONTAINS 0'S WHERE 1'S IN MASK
;       MASK = (2 ** DELAY FACTOR) - 1
;
CLKSYN  TAX                     DELAY FACTOR (=N) TO INDEX
;
RUNO13  LDA     RTCLOK+2        GET LSB OF FRAME COUNTER
        AND     STABLE-1,X      LEAVE N-1 BITS
        BEQ     RUNO13          WAIT FOR NON-ZERO
;
RUNO17  LDA     RTCLOK+2        GET LSB OF FRAME COUNTER AGAIN
        AND     STABLE-1,X      LEAVE N-1 BITS
        BNE     RUNO17          WAIT FOR ZERO
;
        RTS
;
STABLE  .BYTE   $01,$03,$07,$0F,$1F,$3F,$7F
```

```
;START OF LEVEL 4 ROUTINES -- GENERAL PURPOSE UTILITIES

; DECDIG -- CHECK FOR LEGAL DECIMAL DIGIT
; CALLING SEQUENCE:
;       CHARX = CHARACTER IN QUESTION
```

```
;
;  -- START OF LEVEL 4 ROUTINES -- GENERAL PURPOSE UTILITIES
;
;
;  DECDIG -- CHECK FOR LEGAL DECIMAL DIGIT
;
;  CALLING SEQUENCE
;
;       'CHAR' = CHARACTER IN QUESTION
;
;       JSR     DECDIG
;       RCC     NOT A DECIMAL DIGIT
;
DECDIG  LDA     CHAR            IS CHARACTER = DIGIT?
        CMP     #'0
        BCC     DIGRTS          NO
        LDA     #'9             MAYBE
        CMP     CHAR            SET CC FOR EXIT
DIGRTS  RTS                     RETURN WITH CC SET
;
;
;  SETUDF -- SET 'UDEF' ADDRESS IN 'FLINE' POINTER
;
;  CALLING SEQUENCE
;
;       JSR     SETUDF
;
;       'FLINE' = ADDRESS OF 1ST BYTE OF 'UDEF'
;
SETUDF  LDA     MEMLO           'UDEF' STARTS AT BOTTOM OF MEMORY
        STA     FLINE
        LDA     MEMLO+1
        STA     FLINE+1
        RTS
;
;
;  CLRUDF -- CLEAR USER DEFINITION AREA OF DEFINITIONS.
;
;  CALLING SEQUENCE:
;
;       JSR     CLRUDF
;
CLRUDF  LDA     #ALINES         ALWAYS USER DEFINITION AREA BY DELETING ALL LINES
        STA     COUNT
        JSR     SETUDF          SETUP POINTER TO UDEF REGION
;
CLUDIO  LDY     #0
        LDA     #EOL            EOL AT BEGINNING DELETES DEFINITION
        STA     (FLINE),Y
        LDX     #FLINE-OTAB     INCREMENT POINTER
        LDY     #LINESIZ
        JSR     PADDY
        DEC     COUNT           ALL LINES DELETED?
        BNE     CLUDIO          NO
```

```
        LDY     #0              YES -- TERMINATE 'UDEF' AREA
        LDA     #$FF
        STA     (FLINE),Y

        RTS


;
; CLRVDF -- CLEAR USER VARIABLE DEFINTION AREA
;
; CALLING SEQUENCE:
;
;       JSR     CLRVDF
;
CLRVDF  LDA     #' '            (BLANK)
        LDX     #0

CLVOIO  STA     VDEF,X
        INX
        CPX     #VSIZE          DONE?
        BNE     CLVOIO          NO

        LDA     #EOL            TERMINATE AREA WITH EOL
        STA     VDEF-1,X

        LDA     #$FF            YES -- TERMINATE 'VDEF' AREA
        STA     VDEF,X

        RTS
```

```
;  DDCRI -- DOUBLE BYTE DECREMENT
;
;  CALLING SEQUENCE
;
;       X = 'DTAB' INDEX TO DOUBLE-BYTE (LO,HI)
;
;       JSR     DDCRI
;
;       'DTAB'(X) = 'DTAB'(X) - 1
;
DDCRI   LDA     DTAB,X          CHECK FOR BORROW
        BNE     DDCO30          NO BORROW
;
        DEC     DTAB+1,X        BORROW FROM MSB
;
DDCO30  DEC     DTAB,X          DECREMENT LSB
        RTS
;
;
;  DINCI -- DOUBLE BYTE INCREMENT
;
;  CALLING SEQUENCE
;
;       X = 'DTAB' INDEX TO DOUBLE BYTE (LO,HI)
;
;       JSR     DINCI
;
;       'DTAB'(X) = 'DTAB'(X) + 1
;
DINCI   INC     DTAB,X          INCREMENT LSB
        BNE     DINO30          NO CARRY
;
        INC     DTAB+1,X        CARRY TO MSB
;
DINO30  RTS
```

```
;GETCH -- GET CHARACTER
;
;CALLING SEQUENCE:
;
;       'KBIN' = 0 MEANS GET DATA FROM MEMORY; ELSE FROM DEVICE
;       'INPT' = POINTER TO MEMORY INPUT DATA (USED WHEN 'KBIN' = 0)
;       'OUTPT' = POINTER TO MEMORY OUTPUT DATA
;
;       JSR     GETCH
;
;       A = 'CHAR' = CHARACTER OF ATASCII DATA
;       DATA STORED IN OUTPUT BUFFER AS WELL
;       'INPT' & 'OUTPT' INDICES UPDATED AS APPROPRIATE
;
;       Y REGISTER IS CLOBBERED
;
GETCH   LDA     KBIN            KEYBOARD INPUT DESIRED?
        BEQ     GCH010          NO -- GET DATA FROM MEMORY.
;
        JSR     CHIN            YES -- GET DATA FROM DEVICE.
        CMP     #EOL            CHECK FOR PREMATURE TERMINATION.
        BNE     GCH020          NO -- STORE DATA NOW.
;
        LDA     #ECINCL
        JMP     DIRECT          YES - FATAL ERROR.
;
GCH010  LDY     INPT+2          GET INDEX.
        LDA     (INPT),Y        GET MEMORY DATA
        INY
        STY     INPT+2          SAVE NEW INDEX
;
GCH020  STA     CHAR            SAVE CHARACTER IN GLOBAL PLACE
        LDY     OUTPT+1         DON'T STORE IF "BIT-BUCKET"
        CPY     #BUCKET/256
        BEQ     GCH090
;
        LDY     OUTPT+2         GET INDEX.
        CPY     #INSIZ          CHECK FOR LINE OVERFLOW
        BNE     GCH025          NO OVERFLOW
;
        LDA     #ECOLL
        JMP     DIRECT          OVERFLOW -- FATAL ERROR.
;
GCH025  STA     (OUTPT),Y       SAVE DATA TO MEMORY
        INY
        STY     OUTPT+2         SAVE NEW INDEX
;
GCH090  RTS
```

```
; I/O UTILITIES

; ABRTCK -- CHECK FOR ABORT FROM OPERATOR

; CALLING SEQUENCE:

;       JSR     ABRTCK

;       ROUTINE JUMPS TO 'DIRECT' IF ABORTED, ELSE RETURNS

ABRTCK  LDA     BREAK           TEST FOR BREAK KEY
        BNE     ABCOPG          NOT PRESSED.

        LDA     #$FF            PRESSED -- CLEAR FLAG
        STA     BREAK
        LDA     #ECABRT
        JMP     DIRECT          ABORT OPERATION

ABCOPG  RTS


; PRICOM -- PRIORITY COMMAND CHECK

; CALLING SEQUENCE:

;       JSR     PRICOM

; CHECKS FOR PENDING KEYSTROKE FROM KEYBOARD.  IF SO, SUSPENDS
; CURRENT COMMAND AND INITIATES THE ONE PENDING.  AT COMPLETION IT
; RESUMES THE PRIOR COMMAND AND RETURNS.

PRICOM  LDA     EXEC
        BEQ     PRIOPG          NON-EXECUTE

        LDA     CH              KEYSTROKE?
        CMP     #$FF
        BEQ     PRIOPG          NO

        LDA     SPEED           IGNORE IF IN SINGLE-STEP OPERATION
        CMP     #SCSTEP
        BEQ     PRIOPG

        JSR     PUSHIO          PUSH HARDWARE STACK TO SOFTWARE STACK.

        LDX     #PTRSAV+1-DTAB  SAVE KEY DATA
        JSR     SPSHI               ALL POINTERS

        LDX     #LINBIN-DTAB            THE CURRENT COMMAND LINE
        JSR     SPSHI

        LDA     CHAR                    & THE CURRENT COMMAND
        PHA

        JSR     COMMND          GET & EXECUTE ONE COMMAND

        PLA                     RESTORE CURRENT COMMAND ...
        STA     CHAR

        LDX     #LINBIN-DTAB            CURRENT COMMAND LINE
        JSR     SPULI
```

```
            PLA     (PULL)
            JMP     PULLIN          RESTORE HARDWARE STACK FROM SOFTWARE STACK
    RESUME  RTS                     <<< DON'T CHANGE ABOVE JSR TO JMP!!! >>>


    CHIN -- CHARACTER IN FROM CONSOLE

    CALLING SEQUENCE

            JSR     CHIN

            A = CHARACTER

    CHIN    STX     TEMP            SAVE X & Y REGISTERS
            STY     TEMP+1

            LDX     #IOCB1          INPUT IOCB
            JSR     CIO

            CMP     #$1B            CONVERT CTRL-A TO CTRL-Z
            BCS     CHIO10

            CMP     #$01                TO LOWER CASE EQUIVALENT
            BCC     CHIO10

            EOR     #$60

    CHIO10  PHA

        ECHO IF SPLIT MODE

            LDY     MODE
            LDA     TTDY,Y
            CMP     #2              SPLIT SCREEN?
            BNE     CHIO20          NO.

            PLA                     YES -- GET CHARACTER
            PHA
            JSR     COUT2

    CHIO20  PLA
            LDY     TEMP+1          RESTORE X & Y REGISTERS
            LDX     TEMP
            RTS


    COUT -- CHARACTER OUT TO SCREEN DEVICE

    CALLING SEQUENCE

            A = CHARACTER

            JSR     COUT

    COUT    STX     TEMP            SAVE X & Y REGISTERS
            STY     TEMP+1

        *** EXTERNAL ENTRY POINT ***

    COUT2   LDX     #OPUTC-IOCBAS - OUTPUT TO PE -
            JSR     TOHAND

            JMP     IOERR           CHECK FOR I/O ERRORS & RETURN.
```

```
TOUT    STX     TEMP            SAVE X & Y REGISTERS
        STY     TEMP+1

        *** EXTERNAL ENTRY POINT ***

COUTC   LDX     #$PUTC-IOVBAS   OUTPUT TO "E:"
        JSR     IOHAND

        JMP     IOERCK          CHECK FOR I/O ERRORS & RETURN



;TOUT -- TURTLE VALUE OUT TO DISPLAY DEVICE

; CALLING SEQUENCE

;       A = TURTLE CHARACTER

        JSR     TOUT

TTOUT   STX     TEMP            SAVE X & Y REGISTERS
        STY     TEMP+1
        DEC     DSPFLG          INHIBIT CONTROL CHARACTER PROCESSING
        LDX     #$PUTC-IOVBAS   OUTPUT TO "E:"
        JSR     IOHAND
        INC     DSPFLG          RE-ENABLE CONTROL CHARACTER PROCESSING

        JMP     IOERCK          CHECK FOR I/O ERRORS & RETURN



; TIN -- TURTLE VALUE IN FROM DISPLAY DEVICE

; CALLING SEQUENCE

        JSR     TIN

;       A = COLOR VALUE UNDER TURTLE

TIN     STX     TEMP            SAVE X & Y REGISTERS
        STY     TEMP+1
        LDX     #$GETC-IOVBAS   INPUT FROM "S:"
        JSR     IOHAND

    *** EXTERNAL ENTRY POINT ***

IOERCK  CPY     #0              GOOD STATUS?
        BPL     TINDIO          YES

        LDA     #$CIOER
        JMP     DIRECT          NO -- DEBUG ONLY LINE

TINDIO  LDY     TEMP+1          RESTORE X & Y REGISTERS
        LDX     TEMP
        RTS



; IOHAND -- DIVERT I/O INTERFACE ROUTINE

; CALLING SEQUENCE

;       X = I/O ROUTINE OFFSET TO ADDRESS TABLE ENTRY (SYSTEM)

        JSR     IOHAND

        CLOBBERS Y REGISTER
```

```
;
;  DNAME -- READ INPUT FOR DEVICE NAME (" -DEVICE NAME-")
;
;  CALLING SEQUENCE:
;
;       JSR     DNAME
;
;  'OPNBUF' CONTAINS DEVICE SPECIFICATION
;
DNAME   JSR     GETCH           SCAN TO OPENING QUOTE
        CMP     #'"
        BNE     DNAME           KEEP ON SEARCHING
;
        LDA     FF00
        BNE     DNM020          EXECUTE
;
DNM010  JSR     GETCH           NON-EXECUTE -- SCAN TO CLOSING QUOTE
        CMP     #'"
        BNE     DNM010
;
        RTS
;
DNM020  LDX     #0
;
DNM025  JSR     GETCH           GET DEVICE SPECIFICATION
        CMP     #'"             CLOSING QUOTE?
        BEQ     DNM030          YES
;
        STA     OPNBUF,X        NO -- STORE DATA
        INX
        CPX     #DNSIZE+2       CHECK NAME LENGTH
        BNE     DNM025          O.K.
;
        LDA     #ECONTL         DEVICE NAME TOO LONG
        JMP     DIRECT
;
DNM030  LDA     #EOL            TERMINATE NAME IN BUFFER
        STA     OPNBUF,X
;
        RTS
;
;
;
;  DOPEN -- OPEN IOCB3 FOR SPECIFIED DIRECTION
;
;  CALLING SEQUENCE:
;
;       A = OPEN DIRECTION
;
;       JSR     DOPEN
;
;  SETS UP COMMAND BYTE AFTER OPEN FOR GETCH OR PUTCH.
;  DOES NOT RETURN IF OPEN ERROR IS ENCOUNTERED
;
DOPEN   STA     IOCB3+ICAUX1    SAVE OPEN DIRECTION
        LDA     #0
        STA     IOCB3+ICAUX2
        LDX     #$30
```

```
;       SETS UP COMMAND BYTE AFTER OPEN FOR SFETCH OR PUTCH
;       DOES NOT RETURN IF OPEN ERROR IS ENCOUNTERED

DOPEN   STA     DIRECT+1,X      SAVE OPEN DIRECTION
        LDA     #0
        STA     IOCB3+ICAUX2
        LDA     #OPEN
        STA     IOCB3+ICCOM

        LDX     #IOCB3          OPEN DEVICE
        JSR     CIO

        CPY     #0              CHECK STATUS
        BPL     DOPOIO          O.K.

        JSR     DCLOSE          A.D. -- QUIT
        LDA     #ECOPEN         OPEN ERROR CODE
        JMP     DIRECT

DOPOIC  LDA     #GETC           SETUP COMMAND FOR I/O THAT FOLLOWS
        LDY     IOCB3+ICAUX1        BASED ON OPEN DIRECTION
        CPY     #OREAD          ASSUME READ
        BEQ     DOPOCO

        LDA     #PUTC           NO -- WRITE

DOPOCO  STA     IOCB3+ICCOM
        RTS


;  DIN & DOUT -- IOCB3 DATA IN AND OUT

;  CALLING SEQUENCES:

;       A = DATA

        JSR     DOUT

;  OR

;       JSR     DIN

;       A = DATA

DOUT
DIN     STX     TEMP            SAVE X & Y REGISTERS
        STY     TEMP+1

        LDX     #IOCB3          DO I/O OPERATION
        JSR     CIO

        CPY     #0              CHECK STATUS
        BPL     DIODIO          O.K.

        JSR     DCLOSE          ERROR -- CLOSE DEVICE
        LDA     #ECIOER
        JMP     DIRECT

DIODIO  LDY     TEMP+1          RESTORE X & Y REGISTERS
        LDX     TEMP
        RTS


;  DCLOSE -- CLOSE IOCB3

;  CALLING SEQUENCE
```

```
;       JSR     DCLOSE
;
DCLOSE  LDA     #CLOSE          CLOSE DEVICE
        STA     IOCB3+ICCOM

        LDX     #IOCB3
        JSR     CIO

        RTS
```

```
;
; PADDY -- ADD Y TO ADDRESS POINTER
;
; CALLING SEQUENCE
;
;       Y = UNSIGNED NUMBER (0-255)
;       X = 'DTAB' INDEX
;
;       JSR     PADDY
;
;       'DTAB'(X) = 'DTAB'(X) + Y
;
PADDY   CLC
        TYA
        ADC     DTAB,X
        STA     DTAB,X
        BCC     PAD090          NO CARRY -- ALL DONE
;
        INC     DTAB+1,X        CARRY TO MSD
;
PAD090  RTS
```

```
; DTAB UTILITIES -- TAKE UNIT 'DTAB'(X) STRING (RECORD)
;
; DCLR1 -- CLEAR RECORD TO ZEROS
;
; CALLING SEQUENCE:
;
;       X = 'DTAB' INDEX TO RECORD
;
;       JSR     DCLR1
;
;       CLEARS RECORD TO ASCII ZEROS
;
DCLR1   LDA     DTAB-1,X        GET STRING LENGTH
        STA     TEMP
        LDA     #'0             FILL VALUE
;
DCLR10  STA     DTAB,X          STORE A BYTE
        INX
        DEC     TEMP            DONE?
        BNE     DCLR10          NO
;
        RTS                     YES
;
;
; DDCNT -- STRING DECREMENT
;
; CALLING SEQUENCE:
;
;       X = 'DTAB' INDEX TO RECORD
;
;       JSR     DDCR1
;
;       DTAB(X) = DTAB(X) - 1 (UNLESS IT IS ZERO)
;
DDCR1   JSR     TSTNUM          SEE IF NUMBER IS ZERO
        BNE     DDC020          NO -- DO DECREMENT
;
DDC000  RTS                     ALL DONE
;
DDC020  TXA                     CALCULATE INDEX TO END OF STRING
        CLC
        ADC     DTAB-1,X        ADD LENGTH
        TAX                     POINTS TO 1 PAST END OF STRING
;
DDC040  DEC     DTAB-1,X        DECREMENT DIGIT
        LDA     DTAB-1,X        CHECK FOR UNDERFLOW
        CMP     #'0
        BCS     DDC020          B.P.
;
        LDA     #'9             DIGIT UNDERFLOW -- SET TO 9
        STA     DTAB-1,X
        DEX
        JMP     DDC040              & BORROW
;
;
; DINCI -- STRING INCREMENT
;
; CALLING SEQUENCE:
;
;       X = 'DTAB' INDEX TO RECORD
;
;       JSR     DINCI
;
;       (DTAB(X) = DTAB(X) + 1 (UNLESS = ALL 9'S)
```

```
;       X = DTAB  INDEX TO RECORD
;
;       JSR     SINCI
;
;       DTAB(X) = (DTAB(X)-1) ; OVERFLOW = ALL 9'S
;
SINCI   STX     TEMP            SAVE INDEX
        LDA     DTAB-1,X        # OF DIGITS IN NUMBER
        STA     TEMP+1
;
SINOIO  LDA     DTAB,X          CHECK FOR ALL 0'S FIRST
        CMP     #'0
        BNE     SINOSO          NOT ALL 0'S
;
        INX
        DEC     TEMP+1
        BNE     SINOIO          MORE DIGITS TO CHECK
;
                                ALL 0'S -- DON'T INCREMENT
SINORD  RTS
;
SINOSO  LDX     TEMP            RESTORE STARTING INDEX
        TXA                     CALCULATE INDEX TO END OF STRING
        CLC
        ADC     DTAB-1,X        ADD LENGTH
        TAX                     NOW POINTS 1 PAST END OF STRING
;
SINO40  INC     DTAB-1,X        INCREMENT DIGIT
        LDA     #'9             CHECK FOR OVERFLOW
        CMP     DTAB-1,X
        BCS     SINO20          O.K.
;
        LDA     #'0             DIGIT OVERFLOW -- SET TO 0
        STA     DTAB-1,X
        DEX
        BNE     SINO40              & CARRY TO NEXT DIGIT
;
;
;  SMOVI -- MOVE CONTENT OF ONE RECORD TO ANOTHER
;
;  CALLING SEQUENCE:
;
;       X = 'DTAB  INDEX TO SOURCE RECORD
;       Y = 'DTAB' INDEX TO DESTINATION RECORD
;
;       JSR     SMOVI
;
;       'DTAB'(Y) = 'DTAB'(X)
;
SMOVI   LDA     DTAB-1,X        GET RECORD LENGTH FROM SOURCE
        STA     TEMP
;
SMVOIO  LDA     DTAB,X          MOVE DATA FROM SOURCE
        STA     DTAB,Y              TO DESTINATION
        INX
        INY
        DEC     TEMP            DONE?
        BNE     SMVOIO          NO
;
        RTS
;
;
;
;  SPSHI -- PUSH STRING TO STACK
```

```
;  CALLING SEQUENCE:
;
;       X <- 'DTAB' INDEX OF STRING
;
;       JSR     SPUSH
;
;       HARDWARE STACK <- STRING DATA
;
SPUSH   PLA                     REMOVE RETURN ADDRESS FROM STACK TEMPORARILY
        STA     XJUMP+1
        PLA
        STA     XJUMP+2
        LDA     DTAB+1,X        GET STRING LENGTH
        STA     TEMP
;
SPUSLO  LDA     DTAB,X          GET DATA
        PHA                         & PUSH TO STACK
        INX
        DEC     TEMP
        BNE     SPUSLO
;
        JMP     SPLRET          COMMON CODE FOR RETURN
;
;
;  SPULL -- PULL STRING DATA FROM STACK
;
;  CALLING SEQUENCE:
;
;       X <- 'DTAB' INDEX TO STRING
;
;       JSR     SPULL
;
;       DTAB(X) <- DATA FROM STACK
;
SPULL   PLA                     REMOVE RETURN ADDRESS FROM STACK TEMPORARILY
        STA     XJUMP+1
        PLA
        STA     XJUMP+2
        STX     TEMP            SAVE INDEX TO MSD
        TXA                     CALCULATE INDEX TO LSD + 1
        CLC
        ADC     DTAB+1,X            BY ADDING STRING LENGTH TO START INDEX
        TAX
;
SPULLO  DEX
        PLA                     PULL DATA FROM STACK
        STA     DTAB,X              TO 'DTAB'
        CPX     TEMP
        BNE     SPULLO
;
;  EXTERNAL ENTRY POINT
;
SPLRET  LDA     XJUMP+2         RESTORE RETURN ADDRESS TO STACK
        PHA
        LDA     XJUMP+1
        PHA
        RTS                     RETURN
;
;
;
;  SWAPI -- SWAP INDEXED RECORD WITH 'NUMBER'
;
;  CALLING SEQUENCE:
;
;       X = DTAB INDEX OF RECORD
;       'NUMBER' = NUMBER OF STRING RECORDS
```

```
; SWAP1 -- SWAP INDEXED RECORD WITH 'NUMBER'
;
; CALLING SEQUENCE:
;
;       X = DTAB INDEX OF RECORD
;       'NUMBER' = NUMERIC STRING (RECORD)
;
;       JSR     SSWAP1
;
;       'DTAB'(X) AND 'NUMBER' CONTENTS ARE SWAPPED
;
SSWAP1  LDY     NO              SETUP 'NUMBER' INDEX
;
SSW010  LDA     NUMBER,Y        GET 'NUMBER' DATA
        PHA
        LDA     DTAB,X          MOVE 'DTAB' DATA
        STA     NUMBER,Y            TO 'NUMBER'
        PLA
        STA     DTAB,X              & VICE VERSA
        INX
        INY
        CPY     NUMBER+1        DONE?
        BNE     SSW010          NO
;
        RTS
;
;
;
; TSTNUM -- TEST RECORD FOR = ZEROS
;
; CALLING SEQUENCE:
;
;       X = 'DTAB' INDEX TO RECORD
;
;       JSR     TSTNUM
;       BNE     NON-ZERO
;
TSTNUM  STX     TEMP            SAVE DATA INDEX
        LDA     DTAB-1,X        RECORD LENGTH
        STA     TEMP+1
;
TST010  LDA     DTAB,X          GET A DIGIT
        CMP     #'0
        BNE     TST020          NON-ZERO -- CC IS SET FOR EXIT
;
        INX
        DEC     TEMP+1
        BNE     TST010
;
TST020  PHP                     SAVE CC
        LDX     TEMP            RESTORE INDEX
        PLP                     RESTORE CC
        RTS                     RETURN WITH CC SET
;
;
;
; BBTA1 -- CONVERT BINARY BYTE TO ASCII STRING (DECIMAL)
;
; CALLING SEQUENCE:
;
;       A = BINARY NUMBER (UNSIGNED 0-255)
;       X = 'DTAB' INDEX TO RECORD
;
;       JSR     BBTA1
;
;       'DTAB'(X) = ASCII RESULT OF NUMBER CONVERSION
```

```
;  SUBROUTINE TO PERFORM SUBTRACTION OF CONTAINING POWERS OF TEN DECIMAL
;
BTA=A   PHA                     SAVE REGISTERS
        TXA
        PHA
;
        JSR     =CLR1           CLEAR RECORD
;
        PLA                     RESTORE REGISTERS
        TAX
        PLA
;
        LDY     #0              INITIALIZE STRING INDEX
;
SBA010  SEC                     (CLEAR BORROW)
        INC     DTAB+NL-3,Y     PRE-INCREMENT RESULT DIGIT
        SBC     DTATAB,Y        SUBTRACT A POWER OF TEN
        BCS     SBA010          NO UNDERFLOW -- TRY AGAIN
;
        ADC     DTATAB,Y        UNDERFLOW -- UNDO PRE-INCREMENT
        INY
        DEC     DTAB+NL-3,Y         & UNDO UNDERFLOW
        DEY
        CPY     #3              ALL THREE DIGITS?
        BNE     SBA010          NO -- CONTINUE
;
        RTS
;
DTATAB  BYTE    100.10.1        POWERS OF TEN
;
;
;  =ATBI -- CONVERT ASCII STRING (DECIMAL) TO BINARY BYTE
;
;  CALLING SEQUENCE:
;
;       X = DTAB  INDEX TO RECORD
;
;       JSR     =ATBI
;
;       A = BINARY NUMBER (UNSIGNED); CONVERSION OF NUMERIC STRING
;           MODULO 256.
;
;  ALGORITHM:  ((MSD * 10) + NSD) * 10       = LSD
;
=ATBI   LDA     DTAB-1,X        GET STRING LENGTH
        STA     TEMP
;
        LDA     #0              INITIALIZE
;
=ABCD   STA     TEMP+1              RUNNING RESULT
        LDA     DTAB+X          GET AN ASCII DIGIT
        INX
        SEC                     (CLEAR BORROW)
        SBC     #'0             CONVERT TO BCD DIGIT
        CLC
        ADC     TEMP+1          ADD TO PARTIAL RESULT
        DEC     TEMP            MORE DIGITS?
        BEQ     =ABC90          NO -- RESULT IN REGISTER A
;
        ASL     A               YES -- MULTIPLY BY TEN
        STA     TEMP+1          *2
        ASL     A
        ASL     A               *8
        CLC
        ADC     TEMP+1          *10
        JMP     =ABCD           SAVE
```

```
            ASL     A                   X8
            CLC
            ADC     TEMP+1              +10
            JMP     (GOOLU)             SAVE

SAVOUT      RTS


;
;   PUSHHS -- PUSH HARDWARE STACK TO SOFTWARE STACK
;
;   CALLING SEQUENCE:
;
;       JSR     PUSHHS
;
PUSHHS      PLA                         GET RETURN ADDRESS
            STA     XJUMP+1                 & SAVE FOR EXIT
            PLA
            STA     XJUMP+2
            LDY     #3                  INDEX TO SOFTWARE STACK.
;
PHH030      PLA                         MOVE DATA FROM HARDWARE STACK
            STA     (SSTACK),Y              TO SOFTWARE STACK.
            INY
            TSX                         STACK EMPTY?
            CPX     #$FF
            BNE     PHH030              NO.
;
            LDA     SSTACK              NOW STORE STACK FRAME OVERHEAD.
            STA     (SSTACK),Y          OLD FRAME ADDRESS (LO)
            INY
            LDA     SSTACK+1            OLD FRAME ADDRESS (HI)
            STA     (SSTACK),Y
            INY
            TYA                         FRAME INDEX (SIZE + 3).
            SEC                         (CLEAR BORROW)
            SBC     #3
            STA     (SSTACK),Y
            TAY
            INY
            LDX     #SSTACK-DTAB        BUMP POINTER TO END OF NEW FRAME.
            JSR     PADDY
;
            LDA     MEMHI+1             CHECK FOR OVERFLOW ABOUT TO HAPPEN
            CLC                         SET BORROW.
            SBC     SSTACK+1
            BNE     PHH032              NOT WITHIN A PAGE YET -- O.K.
;
            LDA     #SSTKO
            JMP     DIRECT              STACK OVERFLOW -- ABORT.

PHH032      JMP     SPLRET              RETURN TO CALLER


;
;   PULLHS -- PULL DATA FROM SOFTWARE STACK TO HARDWARE STACK
;
;   CALLING SEQUENCE
;
;       JSR     PULLHS
;
PULLHS      PLA                         GET RETURN ADDRESS
            STA     XJUMP                   & SAVE FOR EXIT
```

```
        PLA
        STA     XJUMP+2
        LDY     #0              INDEX TO SOFTWARE STACK.
        LDA     (SSTACK),Y      POINTER ADDRESS (LO).
        PHA                     SAVE TEMPORARILY.
        INY
        LDA     (SSTACK),Y      POINTER ADDRESS (HI).
        PHA                     SAVE TEMPORARILY.
        INY
        LDA     (SSTACK),Y      DATA INDEX (DATA PORTION OF FRAME).
        TAY
        PLA                     POINTER ADDRESS (HI)
        STA     SSTACK+1
        PLA                     POINTER ADDRESS (LO).
        STA     SSTACK

PLH037  LDA     (SSTACK),Y      GET DATA FROM SOFTWARE STACK.
        PHA                     PUSH TO HARDWARE STACK
        DEY                     DECREMENT INDEX.
        CPY     #2              DONE?
        BNE     PLH037          NO.

        JMP     SPLRET          RETURN TO CALLER.
```

```
;  ADDED USER DEFINITIONS FOR 'LOAD' & 'RUN' COMMANDS

LOADTAB  BYTE   'C','Y'          DIFFUSION = Y & HILBERT = J
         WORD   CAN1

         BYTE   'J','Y'          TRINARY TREE = Y & SPIRAL = J
         WORD   CAN2

         BYTE   'J',0            SUPER SPIRAL = A
         WORD   CAN3

         BYTE   'A','Y'          HDS DRAW = Y, SEL DRAW =
         WORD   CAN4

         BYTE   'L','Y'          WALLBANGER = Y, BREAKOUT = J
         WORD   CAN5

         BYTE   'S','J'          HOLLYWOOD SQUARES = J, W/O SQUARE = Y
         WORD   CAN6

         BYTE   'K','Y'          KOCH CURVE = Y
         WORD   CAN7

         BYTE   'B','Y'          THE ZAPPER = Y
         WORD   CAN8

         BYTE   'T','Y'          TURTLE DRAW = Y
         WORD   CAN9

         BYTE   'A','Y'          POSIES = Y
         WORD   CANA

         BYTE   'S','J'          SUPERTURTLE = J
         WORD   CANB

         BYTE   'C',0            COLORPOWER MACHINE
         WORD   CANC

         BYTE   'B','Y'          MAGIC CARPET = Y
         WORD   CAND

         BYTE   $FF              END OF TABLE
         ***=***+32              *** SPARES FOR PATCHING ***


CAN1     BYTE   "1=T[-12F13L03L[2F1+12R"),EOL
         BYTE   "Q=4F",EOL
         BYTE   "Z+T[-V02L23RGJG2LV+12L"),EOL
         BYTE   "V=7[-[DRDVG3L\2RG1+12R",EOL
         BYTE   "V=[%2E4+4(2F1)"),EOL
         BYTE   "J=[XLS+2]",EOL
         BYTE   "E=[UCHM3G2R02G0]",EOL
         BYTE   "BHE_[F9]",EOL
         BYTE   EOF

CAN2     BYTE   "D+[-T]++1-AF2R3(GGR)AFA+1(+4R)!",EOL
         BYTE   "Z=1[2-Z+]_",EOL
         BYTE   "Y=[GHM4+4(+G2R)A+)]",EOL
         BYTE   "[A=(T12-2+2LAF)[_2",EOL
         BYTE   "Y=[D+[Y2M4F)]",EOL
         BYTE   "J=[CHM[CV4ACRR(FOR)HH]",EOL
         BYTE   EOF
```

```
CMD2    BYTE    "$+T;D?F(;T) (!I[_!-#2
        BYTE    "1+!!#P%,#T?!3#!&!F24G!),$!&
        BYTE    "$+!) '$+E&,$???!)",EOL
        BYTE    "/+!3C!(LF!!# !!!T,!!!!_!)",EOL
        BYTE    "&+&!)"&!!",EOL
        BYTE    EOF

CMD1    BYTE    "Y=!H&+?"
        BYTE    C+!"D-CE, 'D
        BYTE    "1!"@YDU$AF_2?$8F_$RRC#_2R#D# !!H!!",EOL
        BYTE    "-#+!"$YDU!O!$)",EOL
        BYTE    "!+!!AF$A#_$C-_$0R_$0L_!",EOL
        BYTE    EOF

CMD3    BYTE    "D+!0R"
        BYTE    CE, 'J,CT, 'J
        BYTE    "1!"$!!?2L8#+!A0 !FM!!",EOL
        BYTE    "Y+!L3T!(3R!!2R0?(!3C!!L3T!(?>L3R!#!!)",EOL
        BYTE    "2+!!DUARFAR)",EOL
        BYTE    "J+!#R#D"
        BYTE    CE, 'J
        BYTE    "&+"MN!!",EOL
        BYTE    EOF

CMD5    BYTE    "V=!(3!!3Y+3-'&0N!)",EOL
        BYTE    "#+!33!3Y+3-'&2$4)",EOL
        BYTE    "Y=!2007RFF?V!!",EOL
        BYTE    "J+!"
        BYTE    CD, 'D,CT, 'D,CE, 'J,CR, 'D
        BYTE    "1!"#N!#13?+_-#I34(#IF3R!!5W)!",EOL
        BYTE    EOF

CMD7    BYTE    "1+T!-?0#L3!PRGE9!3Y@I0GL!4R02+)_",EOL
        BYTE    "0+!F",EOL
        BYTE    "J+#!&FG2R)",EOL
        BYTE    "Y+!"
        BYTE    CD, 'J,CR, 'D,CT, 'D
        BYTE    "33+!NA3L!N3FWDC,!!",EOL
        BYTE    EOF

CMD8    BYTE    "3+!3R3F2R!",EOL
        BYTE    "2+!&0IA+A3!CT!4F0&2-?&3!H1!",EOL
        BYTE    "Y+!"
        BYTE    CR,'0,CA, 'D
        BYTE    "#!(!+?!%!2!)!",EOL
        BYTE    "!+!!!3R3F3L!!3L2F3L)!",EOL
        BYTE    "J+!D?!+#FR-)L1",EOL
        BYTE    "M+!T!(?!AF2R!)FR-!_!",EOL
        BYTE    EOF

CMD9    BYTE    "?+!&A!F8!_$8!F8!_$D!L0!_$C!H5$+_!",EOL
        BYTE    "Y+!"
        BYTE    CR, 'D,CT, 'D
        BYTE    "1!"A!!",EOL
        BYTE    EOF

CMDA    BYTE    "Y+!"
        BYTE    CR,'0,CT, '2
        BYTE    "HN!)-#!!",EOL
        BYTE    "#0!U$AF_$0R_$0L_$C!D1!/_3W)",EOL
        BYTE    "!+8!#!(3F9!3L$F)",EOL
        BYTE    "2+!8!8!3P8A$#!3L$F9RAD!+",EOL
        BYTE    EOF

CMDB    BYTE    "!!&4F",EOL
        BYTE    "Y+!2FRRLF!",EOL
        BYTE    "M+!!!2FLF!",EOL
        BYTE    "Y+!!3R!!",EOL
```

```
            BYTE    "1=(B(E(3F8(AW)3LAP8RAU)*",EOL
            BYTE    EOF

CAND        BYTE    "!!WE",EOL
            BYTE    "V=(2FRFLF)",EOL
            BYTE    "W=(FR2FLP)",EOL
            BYTE    "X=(R3FL)",EOL
            BYTE    "Y=(2RFL2FRF2L)",EOL
            BYTE    "Z=(2F2FLRF2L)",EOL
            BYTE    "R=(@#Z+S-T(D-#J2R)(#J+=#J)1",EOL
            BYTE    "L=(@#Z+T(-=#Z))3+=#J2L)1",EOL
            BYTE    "^D=(HON1(^$AAF_$BAR_$D4L_$GH_2W))",EOL
            BYTE    "F=(@#Z+T(-T(-T(-T(<T(-T_=J)+Y)=X)=U)=U)=U3",EOL
            BYTE    "O=%(%U+%W%X+YA12R)" EOL
            BYTE    EOF

CANC        BYTE    "A=(@&A+=#A%1)",EOL
            BYTE    "B=(@#B+=#H%2)",EOL
            BYTE    "C=(@#A+-=#A%1)",EOL
            BYTE    "D=(@&B+-=#H%2)",EOL
            BYTE    "E=(@#C)+=#C&A)",EOL
            BYTE    "F=(@#D+-=#D&Q)",EOL
            BYTE    "G=(@#C+-=#C&A)",EOL
            BYTE    "H=(@#D+-=#D&Q)",EOL
            BYTE    "M=("
            BYTE    CM, 'Q,CT, 'O
            BYTE    "HCN@=#A=#B=#C=#D+N)",EOL
            BYTE    "N=B(3+P(32(B(APR)+)3R)",EOL
            BYTE    "J=1(^$A+A_$B+B_$C+C_$D+D_"
            BYTE    "$E+E_$F+F_$G+G_$H+H_3W)",EOL
            BYTE    EOF

CAND        BYTE    "J=1(^103!3+!3-??&08!9%2&+10W)",EOL
            BYTE    "Y=(@"
            BYTE    CE, '2
            BYTE    "NP!000(10F+P)"
            BYTE    CE, 'O
            BYTE    "8+PJ)",EOL
            BYTE    EOF

;
;  CONTROL CODE EQUATES FOR CANNED PROGRAMS
;
CA      = $61           CTRL-A
CD      = $64           CTRL-D
CE      = $65           CTRL-E
CM      = $6D           CTRL-M
CT      = $74           CTRL-T
```

```
          IF      BOOT-1
;
; CARTRIDGE OVERHEAD BYTES FOR COLLEEN O.S.
;
*=$BFFA

          .WORD   RESTRT
          .BYTE   $00,$05
          .WORD   INIT
          .ENDIF
          IF      BOOT
PND=*
          .ENDIF


          .END    0
```